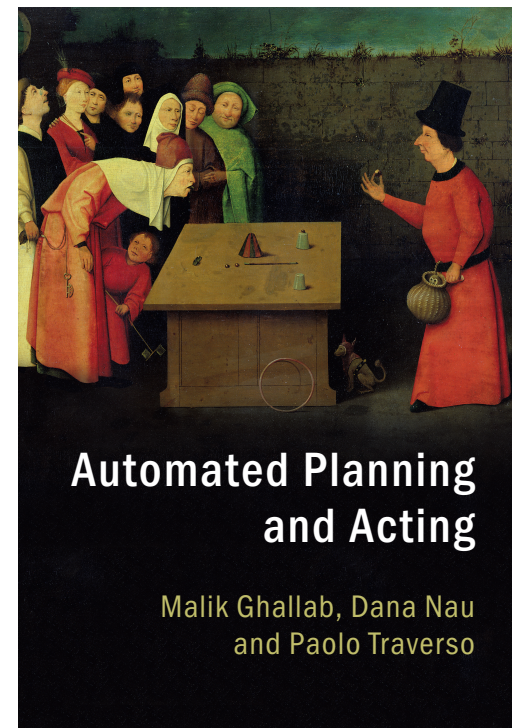


Chapter 6

Deliberation with Probabilistic Domain Models

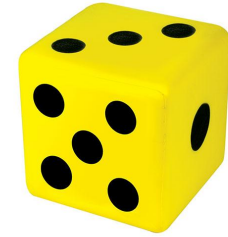
Dana S. Nau
University of Maryland



<http://www.laas.fr/planning>

Motivation

- Situations where actions have multiple possible outcomes and each outcome has a probability
- Several possible action representations
 - Bayes nets, probabilistic actions, ...
- Book doesn't commit to any representation
 - Mainly concentrates on the underlying semantics



roll-die(d)

pre: holding(d) = true

eff:

1/6: top(d) \leftarrow 1

1/6: top(d) \leftarrow 2

1/6: top(d) \leftarrow 3

1/6: top(d) \leftarrow 4

1/6: top(d) \leftarrow 5

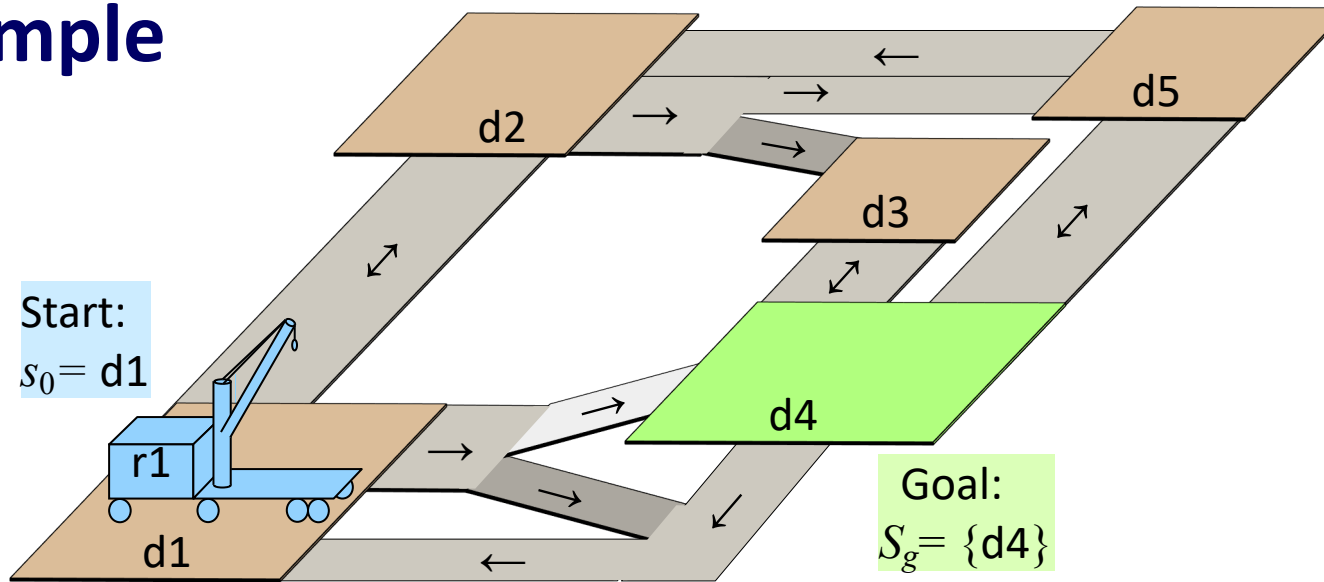
1/6: top(d) \leftarrow 6

Probabilistic Planning Domain

- $\Sigma = (S, A, \gamma, \text{Pr}, \text{cost})$
 - S = set of states
 - A = set of actions
 - $\gamma : S \times A \rightarrow 2^S$
 - $\text{Pr}(s' | s, a)$ = probability of going to state s' if we perform a in s
 - Require $\text{Pr}(s' | s, a) \neq 0$ iff $s' \in \gamma(s, a)$
 - $\text{cost} : S \times A \rightarrow \mathbb{R}_{>0}$
 - $\text{cost}(s, a)$ = cost of action a in state s
 - may omit, default is $\text{cost}(s, a) = 1$

Example

- Robot r1 starts at d1
- Objective: get to d4



- Simplified state names: write $\{loc(r1)=d2\}$ as d2

- Simplified action names: write $move(r1,d2,d3)$ as m23

- r1 has unreliable steering, especially on hills
 - may slip and go elsewhere

$$m12: \quad Pr(d2 \mid d1, m12) = 1$$

m21, m34, m41, m43, m45, m52, m54: like above

$$m14: \quad Pr(d4 \mid d1, m14) = 0.5$$

$$Pr(d1 \mid d1, m14) = 0.5$$

$$m23: \quad Pr(d3 \mid d1, m23) = 0.8$$

$$Pr(d5 \mid d1, m23) = 0.2$$

Policies, Problems, Solutions

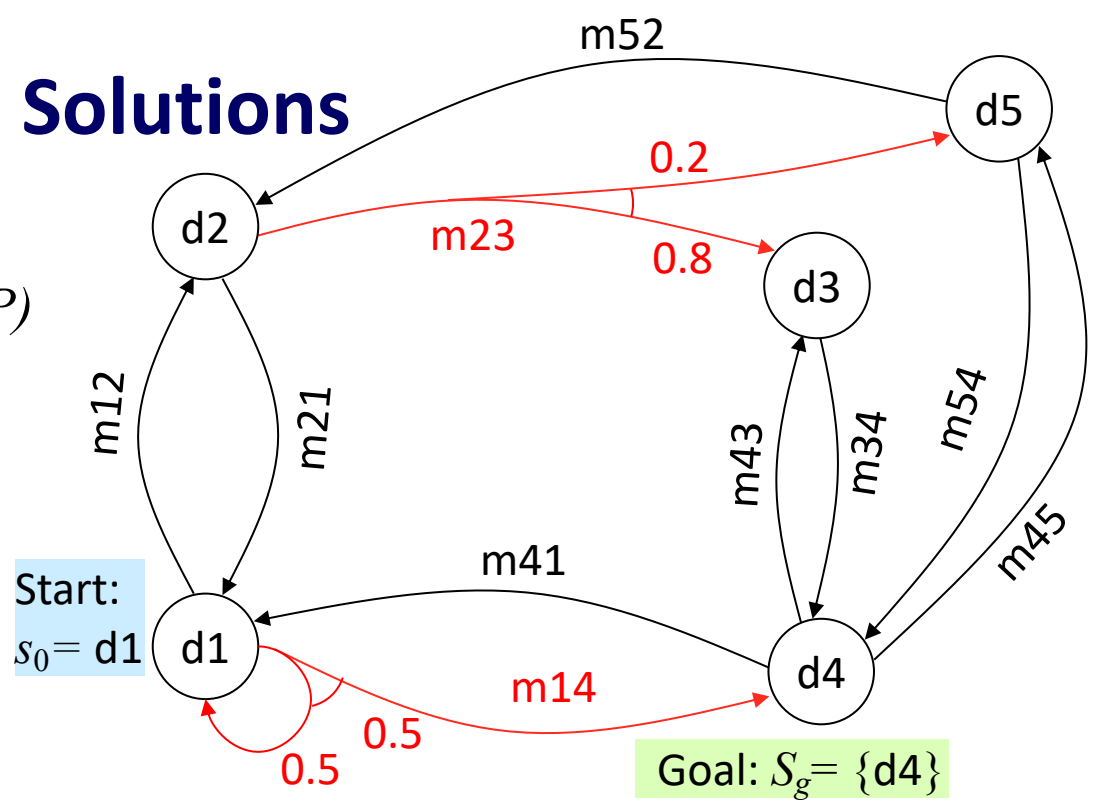
- *Stochastic shortest path (SSP) problem:*

- a triple (Σ, s_0, S_g)

- *Policy:* partial function $\pi : S \rightarrow A$ such that for every $s \in \text{Dom}(\pi) \subseteq S$, $\pi(s) \in \text{Applicable}(s)$

- *Solution* for (Σ, s_0, S_g) : a policy π such that $s_0 \in \text{Dom}(\pi)$ and

- ~~leaves~~ $(s_0, \pi) \cap S_g \neq \emptyset$
 - $\hat{y}(s_0, \pi) \cap S_g \neq \emptyset$



m14: $\Pr(d4 \mid d1, m14) = 0.5$

$\Pr(d1 \mid d1, m14) = 0.5$

m23: $\Pr(d3 \mid d1, m23) = 0.8$

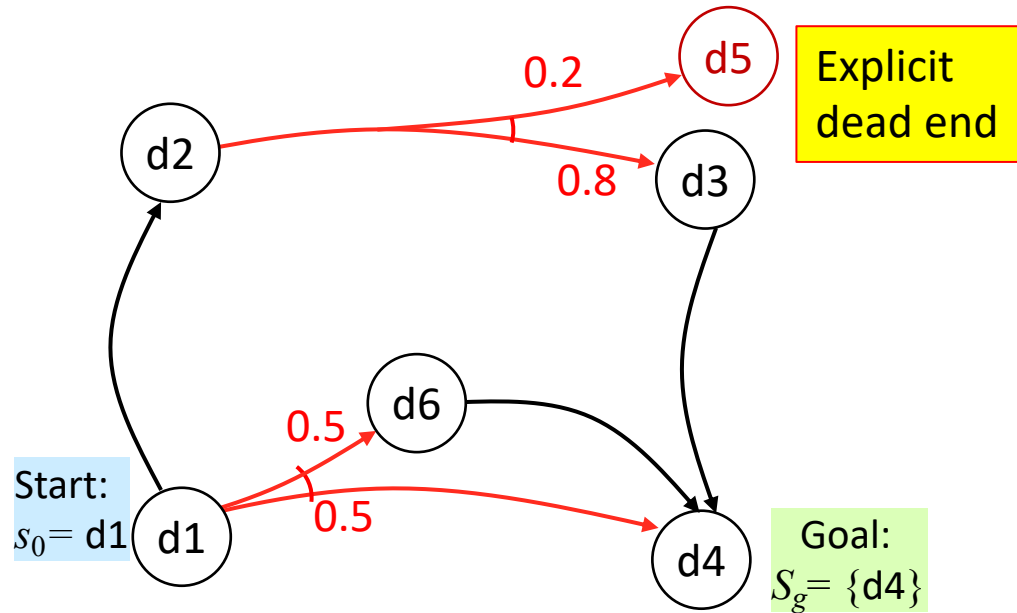
$\Pr(d5 \mid d1, m23) = 0.2$

Notation and Terminology

- *Transitive closure*
 - $\hat{\gamma}(s, \pi) = \{s \text{ and all states reachable from } s \text{ using } \pi\}$
- $\text{Graph}(s, \pi) = \text{rooted graph induced by } \pi \text{ at } s$
 - nodes: $\hat{\gamma}(s, \pi)$
 - edges: state transitions
- $\text{leaves}(s, \pi) = \hat{\gamma}(s, \pi) \setminus \text{Dom}(\pi)$
- A solution policy π is *closed* if it doesn't stop at non-goal states unless there's no way to continue
 - for every state in $\hat{\gamma}(s, \pi)$, either
 - $s \in \text{Dom}(\pi)$ (i.e., π specifies an action at s)
 - $s \in S_g$ (i.e., s is a goal state)
 - $\text{Applicable}(s) = \emptyset$ (i.e., there are no applicable actions at s)

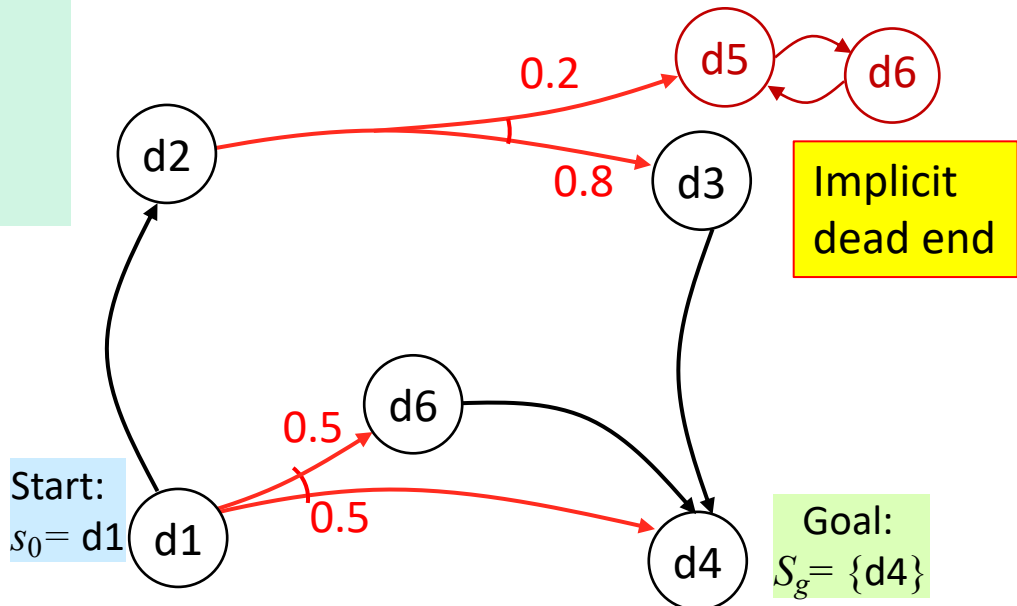
Dead Ends

- Dead end:
 - A state or set of states from which the goal is unreachable



Poll: Would the “dead end” definition from Chapter 5 give us the same dead ends?

1. yes
2. no
3. huh?



Histories

- **History:** sequence of states

$$\sigma = \langle s_0, s_1, s_2, \dots \rangle$$

- May be finite or infinite

$$\sigma = \langle d1, d2, d3, d4 \rangle$$

$$\sigma = \langle d1, d2, d1, d2, \dots \rangle$$

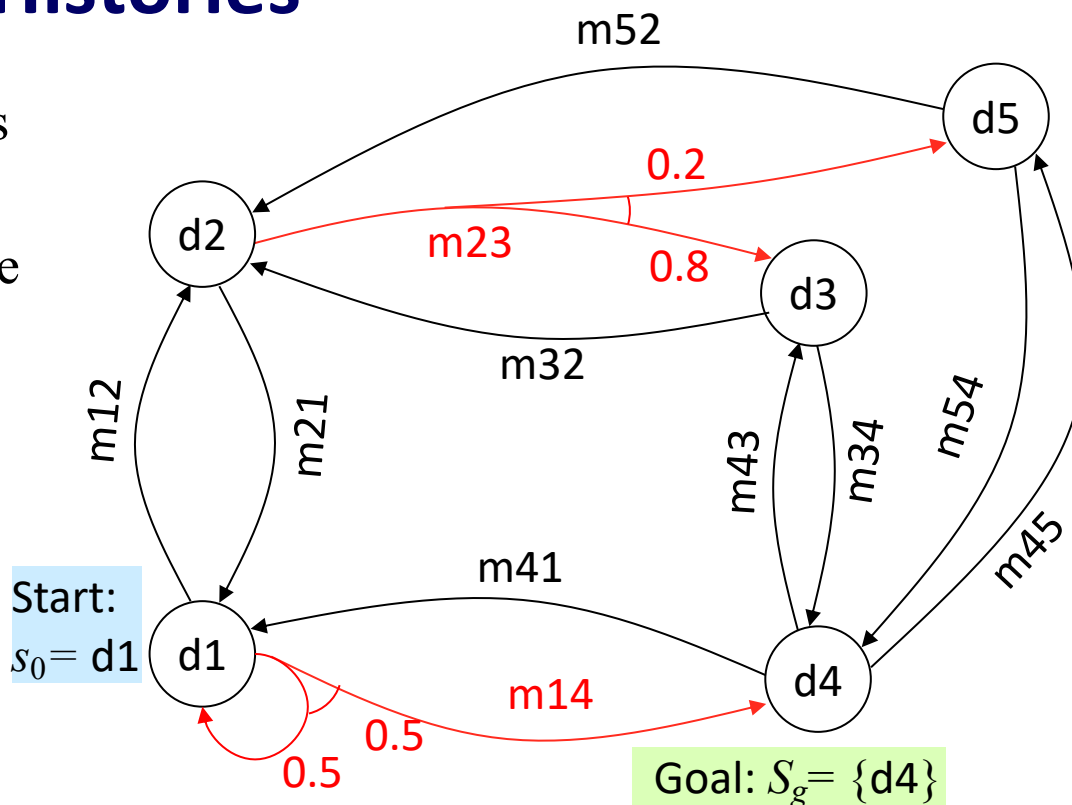
- Let $H(s, \pi) = \{ \text{all possible histories if we start at } s \text{ and follow } \pi, \text{ stopping if } \pi(s) \text{ is undefined or if we reach a goal state} \}$

- If $\sigma \in H(s, \pi)$ then $\Pr(\sigma | s, \pi) = \prod_i \Pr(s_{i+1} | s_i, \pi(s_i))$

- Thus $\sum_{\sigma \in H(s, \pi)} \Pr(\sigma | s, \pi) = 1$

- Probability of reaching a goal state:

- $\Pr(S_g | s, \pi) = \sum_{\sigma \in H(s, \pi)} \{ \Pr(\sigma | s, \pi) \mid \sigma \text{ ends at a state in } S_g \}$



Formula in book is equivalent but more complicated

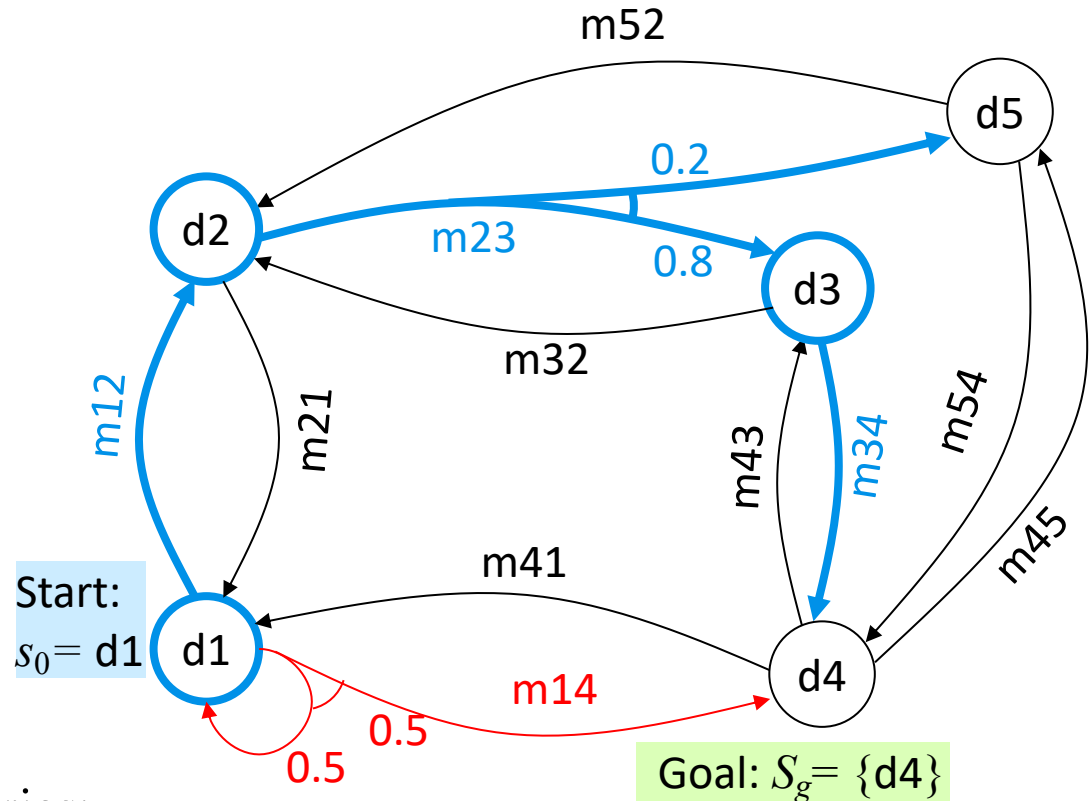
Unsafe Solutions

- Unsafe solution:

- $0 < \Pr(S_g | s_0, \pi) < 1$

- Example:

$$\pi_1 = \{(d1, m12), (d2, m23), (d3, m34)\}$$



- $H(s_0, \pi_1)$ contains two histories:

- $\sigma_1 = \langle d1, d2, d3, d4 \rangle$ $\Pr(\sigma_1 | s_0, \pi_1) = 1 \times .8 \times 1 = .8$

- $\sigma_2 = \langle d1, d2, d5 \rangle$ $\Pr(\sigma_2 | s_0, \pi_1) = 1 \times .2 = .2$

- $\Pr(S_g | s_0, \pi_1) = .8$

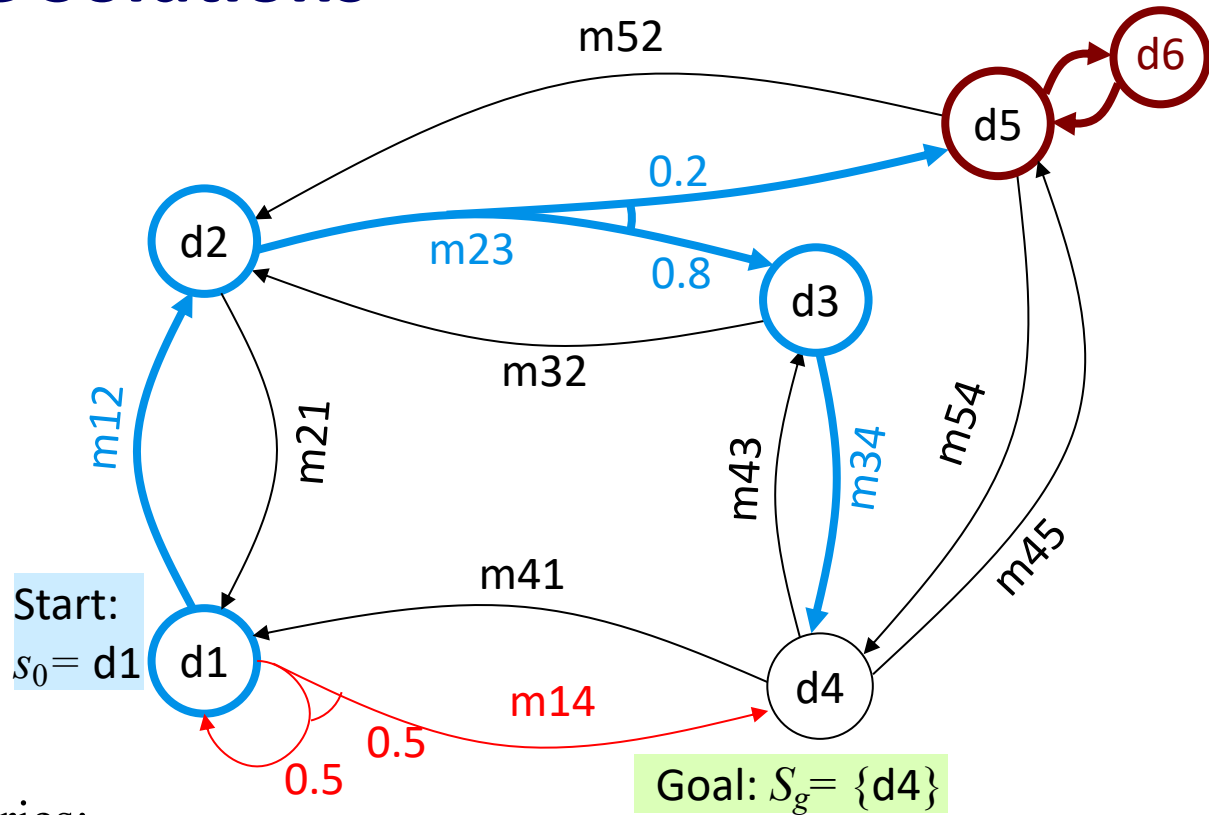
Unsafe Solutions

- Unsafe solution:

➤ $0 < \Pr(S_g | s_0, \pi) < 1$

- Example:

$\pi_2 = \{$
(d1, m12),
(d2, m23),
(d3, m34),
(d5, move(r1,d5,d6)),
(d6, move(r1,d6,d5))\}



- $H(s_0, \pi_2)$ contains two histories:

➤ $\sigma_1 = \langle d1, d2, d3, d4 \rangle$

$\Pr(\sigma_1 | s_0, \pi_2) = 1 \times .8 \times 1 = .8$

➤ $\sigma_3 = \langle d1, d2, d5, d6, d5, d6, \dots \rangle$

$\Pr(\sigma_3 | s_0, \pi_2) = 1 \times .2 \times 1 \times 1 \times 1 \times \dots = .2$

- $\Pr(S_g | s_0, \pi_2) = .8$

Poll: Would the “unsafe solution” definition from Chapter 5 give the same set of policies?
 1. yes 2. no

Safe Solutions

- Safe solution:

- $\Pr(S_g | s_0, \pi) = 1$

- An acyclic safe solution:

$$\pi_3 = \{(d1, m12), (d2, m23), (d3, m34), (d5, m54)\}$$

- $H(s_0, \pi_3)$ contains two histories:

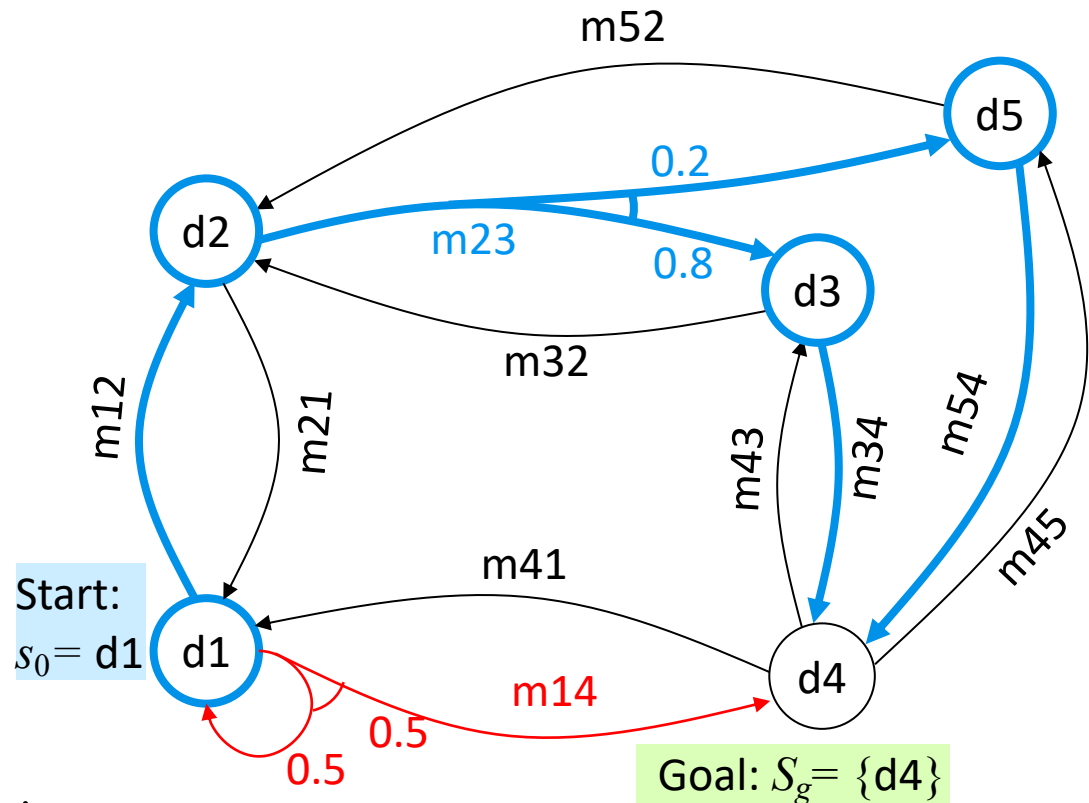
- $\sigma_1 = \langle d1, d2, d3, d4 \rangle$

$$\Pr(\sigma_1 | s_0, \pi_3) = 1 \times .8 \times 1 = .8$$

- $\sigma_4 = \langle d1, d2, d5, d4 \rangle$

$$\Pr(\sigma_4 | s_0, \pi_3) = 1 \times .2 \times 1 = .2$$

$$\Pr(S_g | s_0, \pi_3) = .8 + .2 = 1$$



Safe Solutions

- Safe solution:

- $\Pr(S_g | s_0, \pi) = 1$

- A cyclic safe solution:

- $\pi_4 = \{(d1, m54)\}$

- $H(\pi_4)$ contains infinitely many histories:

- $\sigma_5 = \langle d1, d4 \rangle$

$$\Pr(\sigma_5 | s_0, \pi_4) = 1/2$$

- $\sigma_6 = \langle d1, d1, d4 \rangle$

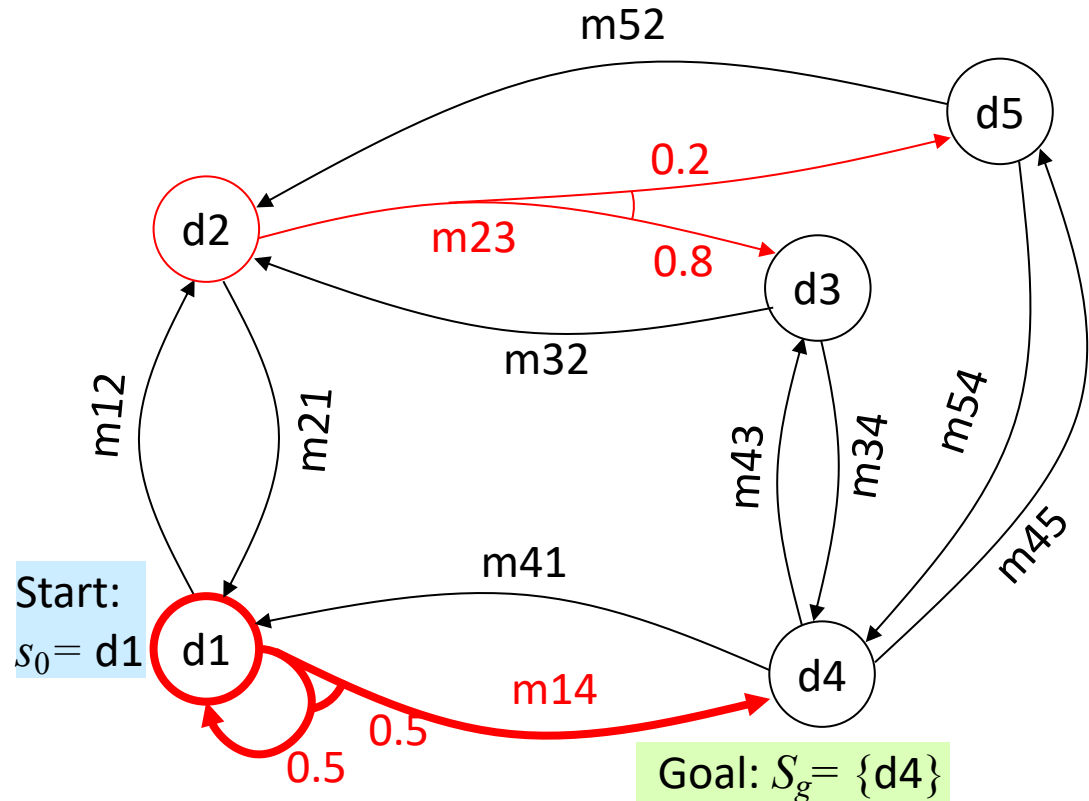
$$\Pr(\sigma_6 | s_0, \pi_4) = (1/2)^2 = 1/4$$

- $\sigma_7 = \langle d1, d1, d1, d4 \rangle$

$$\Pr(\sigma_7 | s_0, \pi_4) = (1/2)^3 = 1/8$$

...

$$\Pr(S_g | s_0, \pi_4) = 1/2 + 1/4 + 1/8 + \dots = 1$$



Safe Solutions

- Safe solution:

- $\Pr(S_g | s_0, \pi) = 1$

- Another cyclic safe solution:

$$\pi_5 = \{(d1, m54), (d4, m41)\}$$

- $H(\pi_5) = H(\pi_4)$:

- $\sigma_5 = \langle d1, d4 \rangle$

$$\Pr(\sigma_5 | s_0, \pi_4) = 1/2$$

- $\sigma_6 = \langle d1, d1, d4 \rangle$

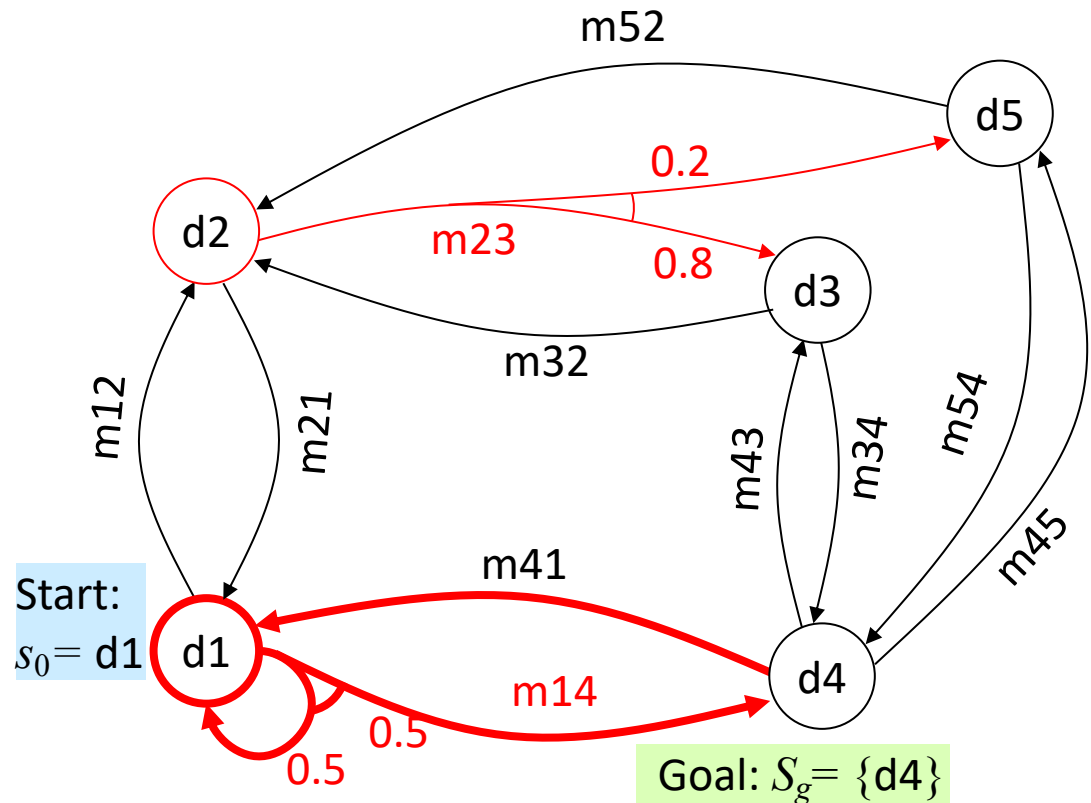
$$\Pr(\sigma_6 | s_0, \pi_4) = (1/2)^2 = 1/4$$

- $\sigma_7 = \langle d1, d1, d1, d4 \rangle$

$$\Pr(\sigma_7 | s_0, \pi_4) = (1/2)^3 = 1/8$$

...

$$\Pr(S_g | s_0, \pi_4) = 1/2 + 1/4 + 1/8 + \dots = 1$$

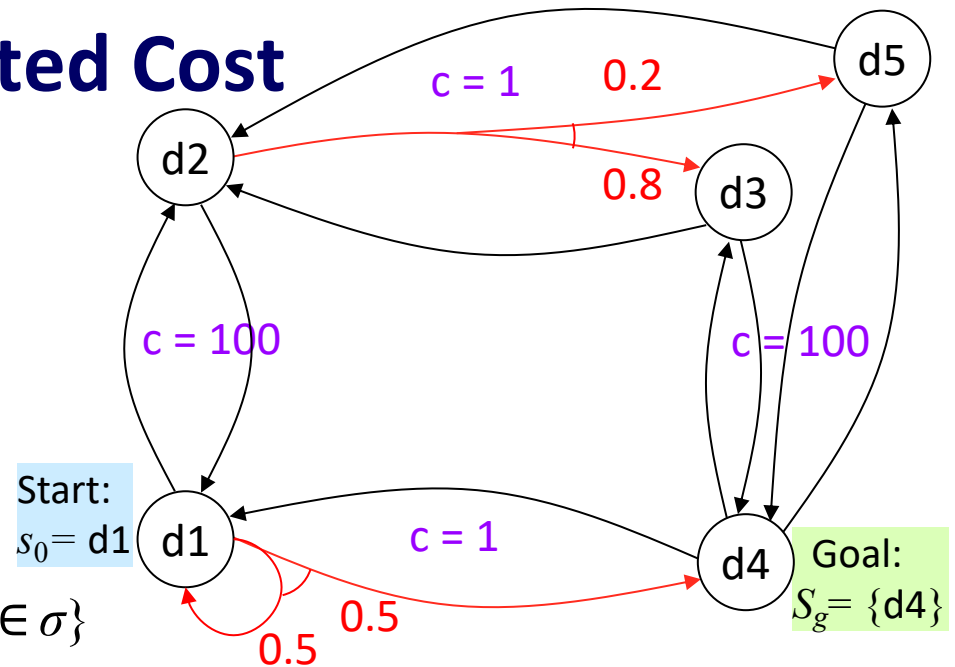


Poll: Would the “safe solution” definition from Chapter 5 give us the same set of policies?
1. yes 2. no

Expected Cost

- $\text{cost}(s, a) = \text{cost of using } a \text{ in } s$
- Example:
 - each “horizontal” action costs 1
 - each “vertical” action costs 100

- History $\sigma = \langle s_0, s_1, s_2, \dots \rangle$
 - $\text{cost}(\sigma | s_0, \pi) = \sum \{ \text{cost}(s_i, \pi(s_i)) \mid s_i \in \sigma \}$



- Let π be a safe solution
- At each state $s \in \text{Dom}(\pi)$, expected cost of following π to goal:

- Weighted sum of history costs:

- $V^\pi(s) = \text{cost}(s, \pi(s)) + \sum_{\sigma \in H(s, \pi)} \text{Pr}(\sigma | s, \pi) \text{cost}(\sigma | s, \pi)$

- Recursive equation

$$V^\pi(s) = \begin{cases} 0, & \text{if } s \text{ is a goal} \\ \text{cost}(s, \pi(s)) + \sum_{s' \in \gamma(s, \pi(s))} \text{Pr}(s' | s, \pi(s)) V^\pi(s'), & \text{otherwise} \end{cases}$$

Poll: Which is correct?

1. weighted sum of history costs
2. recursive equation
3. both
4. neither

Example

- $\pi_3 = \{(d1, m12), (d2, m23), (d3, m34), (d5, m54)\}$

- Weighted sum of history costs:

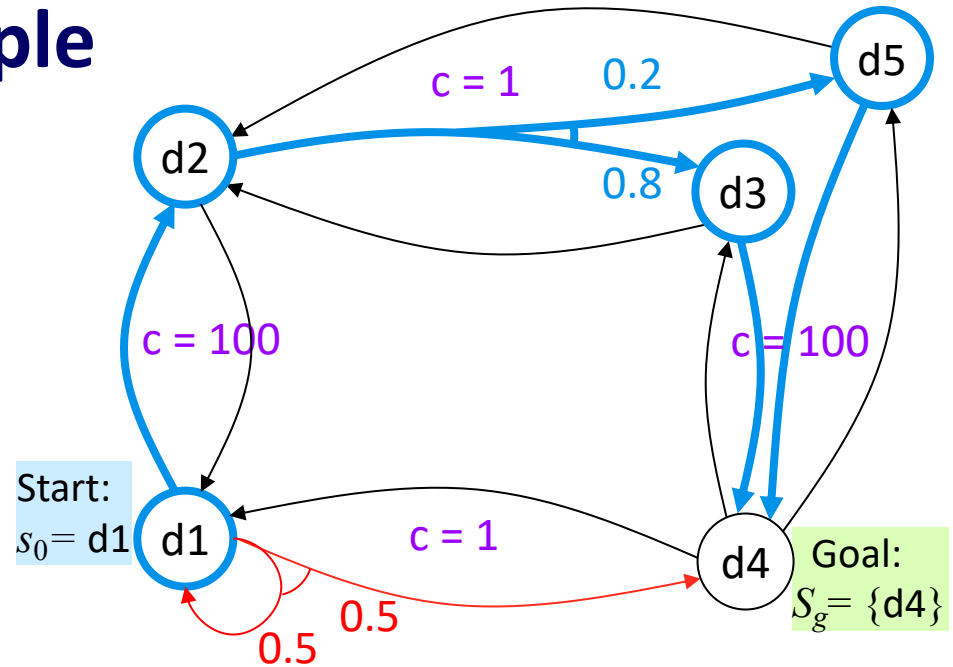
➤ $\sigma_1 = \langle d1, d2, d3, d4 \rangle$

- $\Pr(\sigma_1 | s_0, \pi_3) = 0.8$
- $\text{cost}(\sigma_1 | s_0, \pi_3) = 100 + 1 + 100 = 201$

➤ $\sigma_2 = \langle d1, d2, d5, d4 \rangle$

- $\Pr(\sigma_2 | s_0, \pi_3) = 0.2$
- $\text{cost}(\sigma_2 | s_0, \pi_3) = 100 + 1 + 100 = 201$

- $V^{\pi_1}(d1) = .8(201) + .2(201) = 201$



- Recursive equation:

$$\begin{aligned}
 V^{\pi_1}(d1) &= 100 + V^{\pi_1}(d2) \\
 &= 100 + 1 + .8V^{\pi_1}(d3) + .2V^{\pi_1}(d5) \\
 &= 100 + 1 + .8(100) + .2(100) \\
 &= 201
 \end{aligned}$$

Example

- $\pi_4 = \{(d5, m54)\}$

- Weighted sum of history costs:

- $\sigma_5 = \langle d1, d4 \rangle$

- $\Pr(\sigma_5 | \pi_4) = 1/2$

- $\text{cost}(\sigma_5 | \pi_4) = 1$

- $\sigma_6 = \langle d1, d1, d4 \rangle$

- $\Pr(\sigma_6 | \pi_4) = (1/2)^2$

- $\text{cost}(\sigma_6 | \pi_4) = 2$

- $\sigma_7 = \langle d1, d1, d1, d4 \rangle$

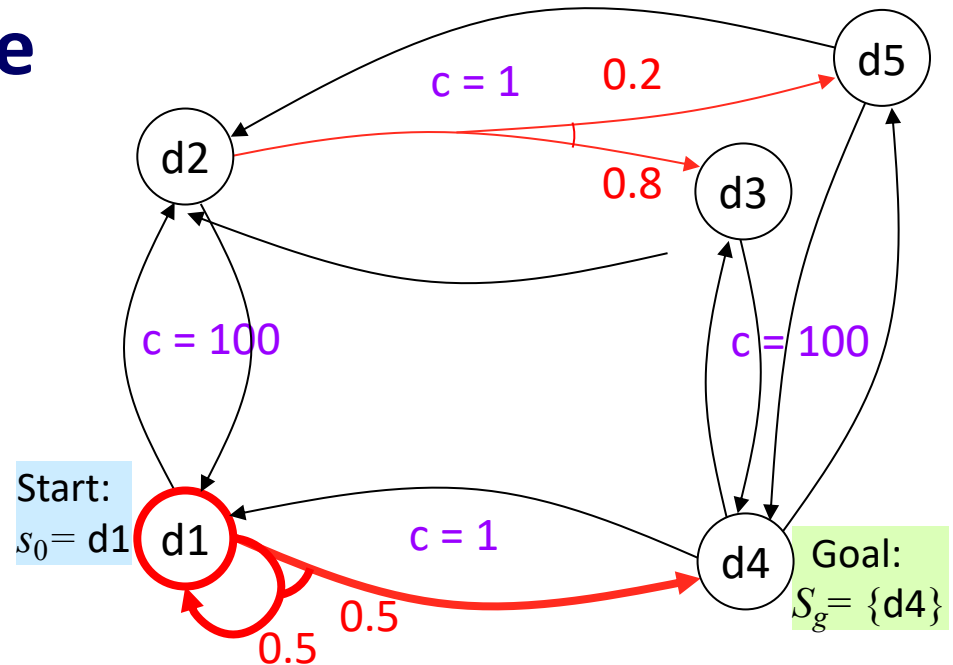
- $\Pr(\sigma_7 | \pi_4) = (1/2)^3$

- $\text{cost}(\sigma_7 | \pi_4) = 3$

...

- $$V^{\pi_4}(d1) = (1/2)1 + (1/2)^2 2 + (1/2)^3 3 + \dots$$

$$= 2$$



- Recursive equation:

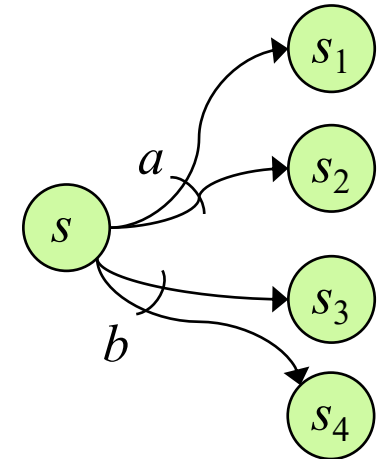
$$V^{\pi_4}(d1) = 1 + 1/2(0) + 1/2(V^{\pi_4}(d1))$$

$$1/2 V^{\pi_4}(d1) = 1$$

$$V^{\pi_4}(d1) = 2$$

Planning as Optimization

- Let π and π' be safe solutions
 - π dominates π' if $V^\pi(s) \leq V^{\pi'}(s)$ for every $s \in \text{Dom}(\pi) \cap \text{Dom}(\pi')$
- π is *optimal* if π dominates every safe solution
 - If π and π' are both optimal, then $V^\pi(s) = V^{\pi'}(s)$ at every state where they're both defined



- $V^*(s)$ = expected cost of getting to goal using an optimal safe solution
- Recall that
$$V^\pi(s) = \begin{cases} 0, & \text{if } s \text{ is a goal} \\ \text{cost}(s, \pi(s)) + \sum_{s' \in \gamma(s, \pi(s))} \text{Pr}(s' | s, \pi(s)) V^\pi(s'), & \text{otherwise} \end{cases}$$

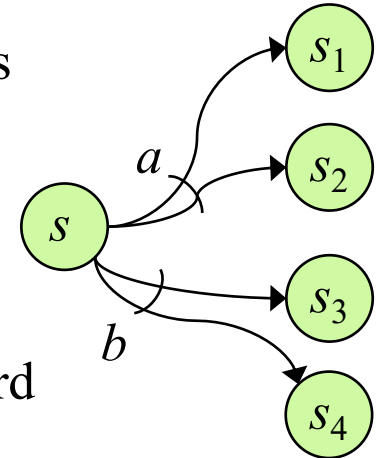
- *Optimality principle* (Bellman's theorem):

$$V^*(s) = \begin{cases} 0, & \text{if } s \text{ is a goal} \\ \min_{a \in \text{Applicable}(s)} \{ \text{cost}(s, a) + \sum_{s' \in \gamma(s, a)} \text{Pr}(s' | s, a) V^*(s') \}, & \text{otherwise} \end{cases}$$

- Intuition: consider what would happen if $V^*(s) \neq \min_{a \in \text{Applicable}(s)} \{ \dots \}$

Cost to Go

- Let (Σ, s_0, S_g) be a *safe* SSP
 - i.e., S_g is reachable from every state
 - same as *safely explorable* in Chapter 5
- Let π be a safe solution that's defined at all non-goal states
 - i.e., $\text{Dom}(\pi) = S \setminus S_g$
- Let $a \in \text{Applicable}(s)$
- *Cost-to-go*:
 - Expected cost if we start at s , use a , and use π afterward
 - $Q^\pi(s, a) = \text{cost}(s, a) + \sum_{s' \in \gamma(s, a)} \text{Pr}(s' | s, a) V^\pi(s')$
- For every $s \in S \setminus S_g$, let $\pi'(s) \in \text{argmin}_{a \in \text{Applicable}(s)} Q^\pi(s, a)$

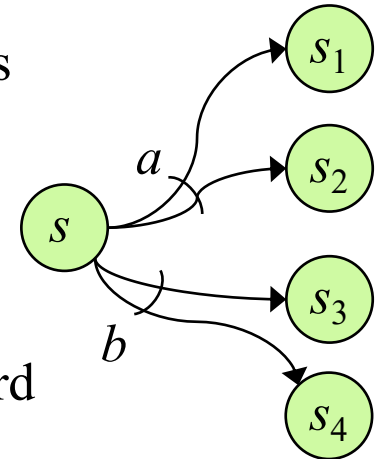


Poll: Is π' a safe solution?

1. yes
2. no

Cost to Go

- Let (Σ, s_0, S_g) be a *safe* SSP
 - i.e., S_g is reachable from every state
 - same as *safely explorable* in Chapter 5
- Let π be a safe solution that's defined at all non-goal states
 - i.e., $\text{Dom}(\pi) = S \setminus S_g$
- Let $a \in \text{Applicable}(s)$
- *Cost-to-go*:
 - Expected cost if we start at s , use a , and use π afterward
 - $Q^\pi(s, a) = \text{cost}(s, a) + \sum_{s' \in \gamma(s, a)} \Pr(s' | s, a) V^\pi(s')$
- For every $s \in S \setminus S_g$, let $\pi'(s) \in \text{argmin}_{a \in \text{Applicable}(s)} Q^\pi(s, a)$



Poll: Which of the following is true?

1. π' dominates π
2. π dominates π'
3. both
4. neither

Policy Iteration

- $PI(\Sigma, s_0, S_g, \pi_0)$

$\pi \leftarrow \pi_0$

loop

compute $\{V^\pi(s) \mid s \in S\}$

for every non-goal state s do

$A \leftarrow \operatorname{argmin}_{a \in \text{Applicable}(s)} Q^\pi(s, a)$

if $\pi(s) \in A$ then $\pi'(s) \leftarrow \pi(s)$

else $\pi'(s) \leftarrow$ any action in A

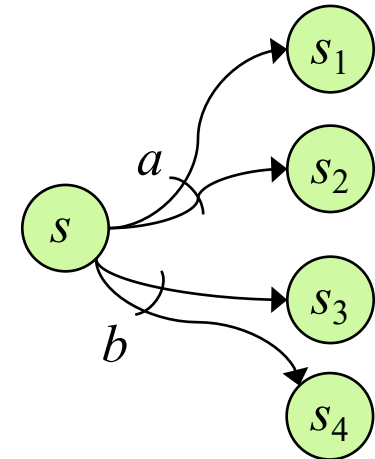
if $\pi' = \pi$ then

return π

$\pi \leftarrow \pi'$

n equations, n unknowns, where $n = |S|$

E (cost of using a then π)

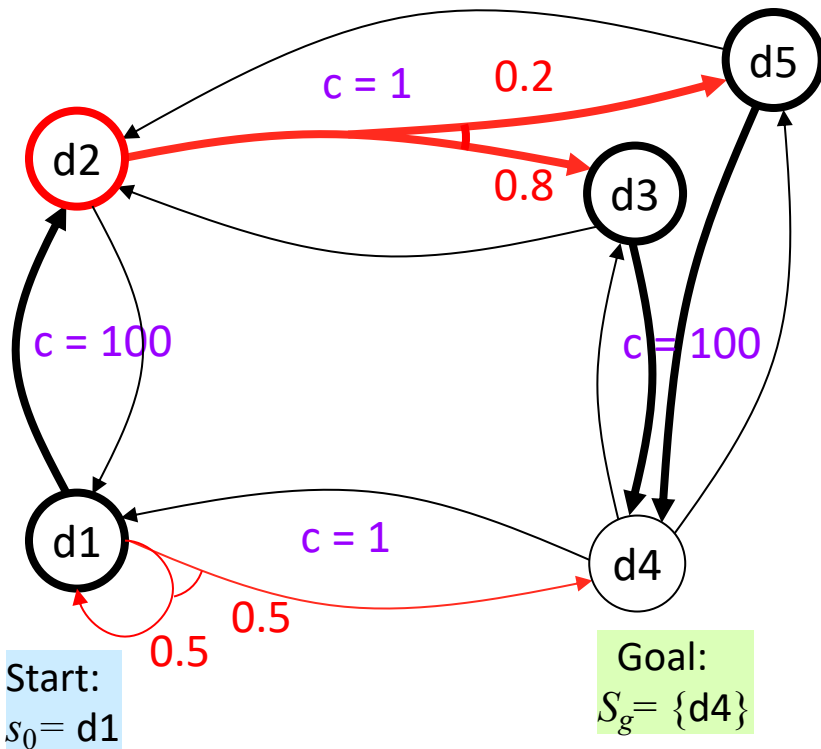


- Converges in a finite number of iterations

Example

Start with

$$\pi = \pi_0 = \{(d1, m12), (d2, m23), (d3, m34), (d5, m54)\}$$



$$V^\pi(d4) = 0$$

$$V^\pi(d3) = 100 + V^\pi(d4) = 100$$

$$V^\pi(d5) = 100 + V^\pi(d5) = 100$$

$$V^\pi(d2) = 1 + (0.8 V^\pi(d3) + 0.2 V^\pi(d5)) = 101$$

$$V^\pi(d1) = 100 + V^\pi(d2) = 201$$

$$Q(d1, m12) = 100 + 101 = 201$$

$$Q(d1, m14) = 1 + \frac{1}{2} \times 201 + \frac{1}{2}(0) = 101.5$$

$$\text{argmin} = m14$$

$$Q(d2, m23) = 1 + (0.8(100) + 0.2(100)) = 101$$

$$Q(d2, m21) = 100 + 201 = 301$$

$$\text{argmin} = m23$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$Q(d3, \text{move}(r1, d3, d2)) = 100 + 101 = 201$$

$$\text{argmin} = m34$$

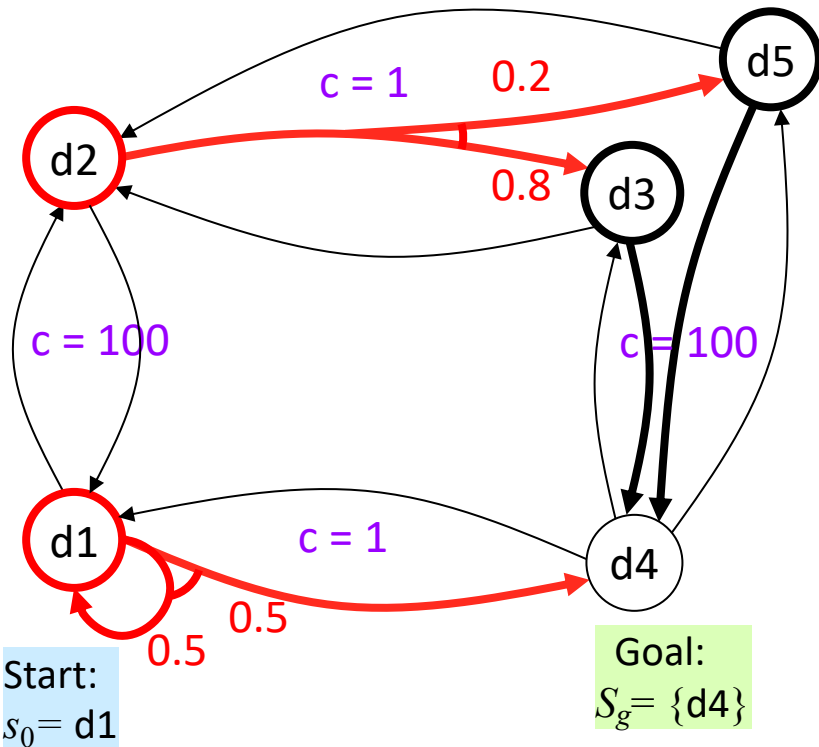
$$Q(d5, m54) = 100 + 0 = 100$$

$$Q(d5, m54) = 100 + 101 = 201$$

$$\text{argmin} = m54$$

Example

$\pi = \{(d1, m14),$
 $(d2, m23),$
 $(d3, m34),$
 $(d5, m54)\}$



$$V^\pi(d4) = 0$$

$$V^\pi(d3) = 100 + V^\pi(d4) = 100$$

$$V^\pi(d5) = 100 + V^\pi(d5) = 100$$

$$V^\pi(d2) = 1 + (0.8 V^\pi(d3) + 0.2 V^\pi(d5)) = 101$$

$$V^\pi(d1) = 1 + \frac{1}{2}V^\pi(d1) + \frac{1}{2}V^\pi(d4) \Rightarrow V^\pi(d1) = 2$$

$$Q(d1, m12) = 100 + 101 = 201$$

$$Q(d1, m14) = 1 + \frac{1}{2}(2) + \frac{1}{2}(0) = 2$$

$$\operatorname{argmin} = m14$$

$$Q(d2, m23) = 1 + (0.8(100) + 0.2(100)) = 101$$

$$Q(d2, m21) = 100 + 2 = 102$$

$$\operatorname{argmin} = m23$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$Q(d3, \text{move}(r1, d3, d2)) = 100 + 101 = 201$$

$$\operatorname{argmin} = m34$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$Q(d5, m54) = 100 + 101 = 201$$

$$\operatorname{argmin} = m54$$

Value Iteration

- Synchronous version (easier to understand)

```
VI( $\Sigma, s_0, S_g, V_0$ )
  for  $i = 1, 2, \dots$ 
    for every nongoal state  $s$ 
      for every applicable action  $a$  do
         $Q(s, a) \leftarrow \text{cost}(s, a) + \sum_{s' \in S} \text{Pr}(s'|s, a) V_{i-1}(s')$ 
         $V_i(s) \leftarrow \min_{a \in \text{Applicable}(s)} Q(s, a)$ 
         $\pi_i(s) \leftarrow \text{argmin}_{a \in \text{Applicable}(s)} Q(s, a)$ 
      if  $\max_{s \in S} |V_i(s) - V_{i-1}(s)| \leq \eta$  then return  $\pi'$ 
```

- $\eta > 0$: for testing approximate convergence
- V_0 is a heuristic function
 - must have $V_0(s) = 0$ for every $s \in S_g$
 - e.g., adapt a heuristic from Chapter 2
- V_i = values computed at i 'th iteration
- π_i = plan computed from V_i

- Asynchronous version (more efficient)

```
VI( $\Sigma, s_0, S_g, V_0$ )
  global  $\pi \leftarrow \emptyset$ ; global  $V(s) \leftarrow V_0(s) \forall s$ 
  loop
     $r \leftarrow \max_{s \in S \setminus S_g} \text{Bellman-Update}(s)$ 
    if  $r \leq \eta$  then return  $\pi$ 

Bellman-Update( $s$ )
   $v_{\text{old}} \leftarrow V(s)$ 
  for every  $a \in \text{Applicable}(s)$  do
     $Q(s, a) \leftarrow \text{cost}(s, a) + \sum_{s' \in S} \text{Pr}(s'|s, a) V(s')$ 
   $V(s) \leftarrow \min_{a \in \text{Applicable}(s)} Q(s, a)$ 
   $\pi(s) \leftarrow \text{argmin}_{a \in \text{Applicable}(s)} Q(s, a)$ 
  return  $|V(s) - v_{\text{old}}|$ 
```

- Synchronous version computes V_i and π_i from old V_{i-1} and π_{i-1}
- Asynchronous version updates V and π in place; new values available immediately

Synchronous

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14)$$

$$= 1 + (\frac{1}{2}(0) + \frac{1}{2}(0)) = 1$$

$$V_1(d1) = 1; \pi_1(d1) = m14$$

$$Q(d2, m21) = 100 + 0 = 100$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(0) + \frac{1}{2}(0)) = 1$$

$$V_1(d2) = 1; \pi_1(d2) = m23$$

$$Q(d3, m32) = 1 + 0 = 1$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$V_1(d3) = 1; \pi_1(d3) = m32$$

$$Q(d5, m52) = 1 + 0 = 1$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V_1(d5) = 1; \pi_1(d5) = m52$$

$$r = \max(1 - 0, 1 - 0, 1 - 0, 1 - 0) = 1$$

$$\eta = 0.2$$

$$V_0(s) = 0 \text{ for all } s$$

Asynchronous

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14)$$

$$= 1 + (\frac{1}{2}(0) + \frac{1}{2}(0)) = 1$$

$$V(d1) = 1; \pi(d1) = m14$$

$$Q(d2, m21) = 100 + 1 = 101$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(0) + \frac{1}{2}(0)) = 1$$

$$V(d2) = 1; \pi(d2) = m23$$

$$Q(d3, m32) = 1 + 1 = 2$$

$$Q(d3, m34) = 100 + 0 = 100$$

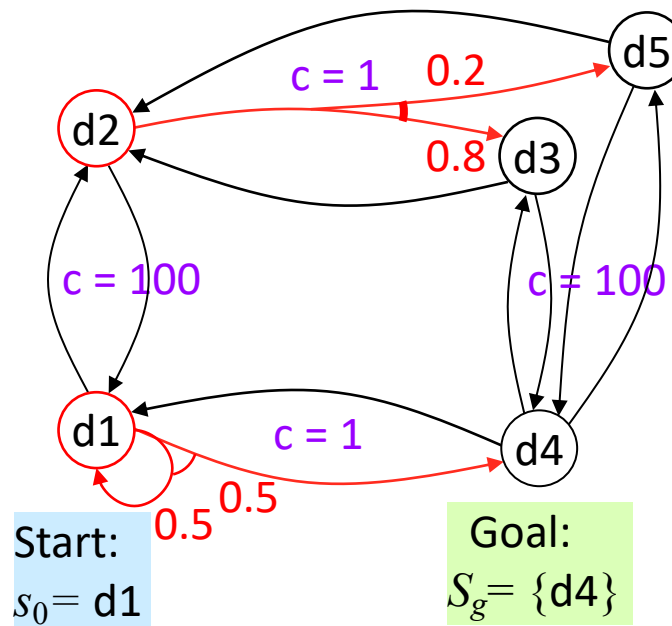
$$V(d3) = 2; \pi(d3) = m32$$

$$Q(d5, m52) = 1 + 1 = 2$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V(d5) = 2; \pi(d5) = m52$$

$$r = \max(1 - 0, 1 - 0, 2 - 0, 2 - 0) = 1$$



Synchronous

$$Q(d1, m12) = 100 + 1 = 101$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1) + \frac{1}{2}(0)) = 1\frac{1}{2}$$

$$V_2(d1) = 1\frac{1}{2}; \pi_2(d1) = m14$$

$$Q(d2, m21) = 100 + 1 = 101$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(1) + \frac{1}{2}(1)) = 2$$

$$V_2(d2) = 2; \pi_2(d2) = m23$$

$$Q(d3, m32) = 1 + 1 = 2$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$V_2(d3) = 2; \pi_2(d3) = m32$$

$$Q(d5, m52) = 1 + 1 = 2$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V_2(d5) = 2; \pi_2(d5) = m52$$

$$r = \max(1\frac{1}{2} - 1, 2 - 1, 2 - 1, 2 - 1) = 1$$

$$\eta = 0.2$$

$$V(d1) = 1$$

$$V(d2) = 1$$

$$V(d3) = 1$$

$$V(d5) = 1$$

$$\eta = 0.2$$

$$V(d1) = 1$$

$$V(d2) = 1$$

$$V(d3) = 2$$

$$V(d5) = 2$$

Asynchronous

$$Q(d1, m12) = 100 + 0 = 101$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1) + \frac{1}{2}(0)) = 1\frac{1}{2}$$

$$V(d1) = 1; \pi(d1) = m14$$

$$Q(d2, m21) = 100 + 1\frac{1}{2} = 101\frac{1}{2}$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(2) + \frac{1}{2}(2)) = 3$$

$$V(d2) = 3; \pi(d2) = m23$$

$$Q(d3, m32) = 1 + 3 = 4$$

$$Q(d3, m34) = 100 + 0 = 100$$

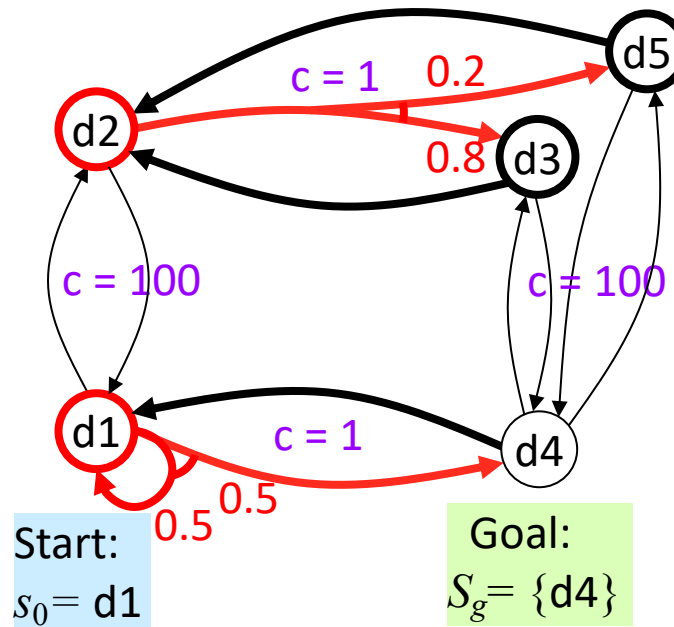
$$V(d3) = 4; \pi(d3) = m32$$

$$Q(d5, m52) = 1 + 3 = 4$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V(d5) = 4; \pi(d5) = m52$$

$$r = \max(1\frac{1}{2} - 1, 3 - 1, 4 - 2, 4 - 2) = 2$$



Synchronous

$$Q(d1, m12) = 100 + 2 = 102$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1\frac{1}{2}) + \frac{1}{2}(0)) = 1\frac{3}{4}$$

$$V_3(d1) = 1\frac{3}{4}; \pi_3(d1) = m14$$

$$Q(d2, m21) = 100 + 1\frac{1}{2} = 101\frac{1}{2}$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(2) + \frac{1}{2}(2)) = 3$$

$$V_3(d2) = 3; \pi_3(d2) = m23$$

$$Q(d3, m32) = 1 + 2 = 3$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$V_3(d3) = 3; \pi_3(d3) = m32$$

$$Q(d5, m52) = 1 + 2 = 3$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V_3(d5) = 3; \pi_3(d5) = m52$$

$$r = \max(1\frac{3}{4} - 1\frac{1}{2}, 3 - 2, 3 - 2, 3 - 2) = 1$$

$$\eta = 0.2$$

$$V(d1) = 1\frac{1}{2}$$

$$V(d2) = 2$$

$$V(d3) = 2$$

$$V(d5) = 2$$

$$\eta = 0.2$$

$$V(d1) = 1\frac{1}{2}$$

$$V(d2) = 3$$

$$V(d3) = 4$$

$$V(d5) = 4$$

Asynchronous

$$Q(d1, m12) = 100 + 3 = 103$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1\frac{1}{2}) + \frac{1}{2}(0)) = 1\frac{3}{4}$$

$$V(d1) = 1\frac{3}{4}; \pi(d1) = m14$$

$$Q(d2, m21) = 100 + 1\frac{3}{4} = 101\frac{3}{4}$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(4) + \frac{1}{2}(4)) = 5$$

$$V(d2) = 5; \pi(d2) = m23$$

$$Q(d3, m32) = 1 + 5 = 6$$

$$Q(d3, m34) = 100 + 0 = 100$$

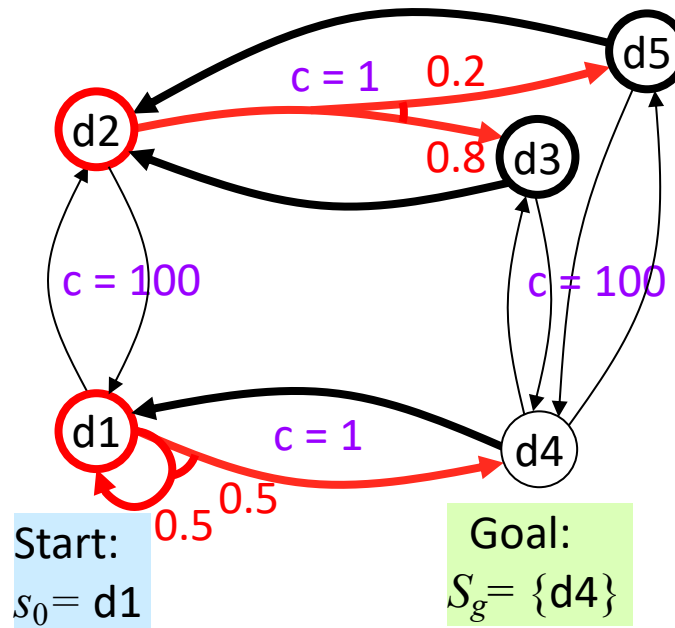
$$V(d3) = 6; \pi(d3) = m32$$

$$Q(d5, m52) = 1 + 5 = 6$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V(d5) = 6; \pi(d5) = m52$$

$$r = \max(1\frac{3}{4} - 1\frac{1}{2}, 5 - 3, 6 - 4, 6 - 4) = 2$$



Synchronous

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1^{3/4}) + \frac{1}{2}(0)) = 1^{7/8}$$

$$V_4(d1) = 1^{7/8}; \pi_4(d1) = m14$$

$$Q(d2, m21) = 100 + 1^{3/4} = 101^{3/4}$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(3) + \frac{1}{2}(3)) = 4$$

$$V_4(d2) = 4; \pi_4(d2) = m23$$

$$Q(d3, m32) = 1 + 3 = 4$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$V_4(d3) = 4; \pi_4(d3) = m32$$

$$Q(d5, m52) = 1 + 3 = 4$$

$$Q(d5, m54) = 100 + 0 = 100$$

$$V_4(d5) = 4; \pi_4(d5) = m52$$

$$r = \max(1^{7/8} - 1^{3/4}, 3 - 2, 3 - 2, 3 - 2) = 1$$

$$\eta = 0.2$$

$$V(d1) = 1^{3/4}$$

$$V(d2) = 3$$

$$V(d3) = 3$$

$$V(d5) = 3$$

$$\eta = 0.2$$

$$V(d1) = 1^{3/4}$$

$$V(d2) = 5$$

$$V(d3) = 6$$

$$V(d5) = 6$$

Asynchronous

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1^{3/4}) + \frac{1}{2}(0)) = 1^{7/8}$$

$$V(d1) = 1^{7/8}; \pi(d1) = m14$$

$$Q(d2, m21) = 100 + 1^{7/8} = 101^{7/8}$$

$$Q(d2, m23) = 1 + (\frac{1}{2}(6) + \frac{1}{2}(6)) = 7$$

$$V(d2) = 7; \pi(d2) = m23$$

$$Q(d3, m32) = 1 + 7 = 8$$

$$Q(d3, m34) = 100 + 0 = 100$$

$$V(d3) = 8; \pi(d3) = m32$$

$$Q(d5, m52) = 1 + 7 = 8$$

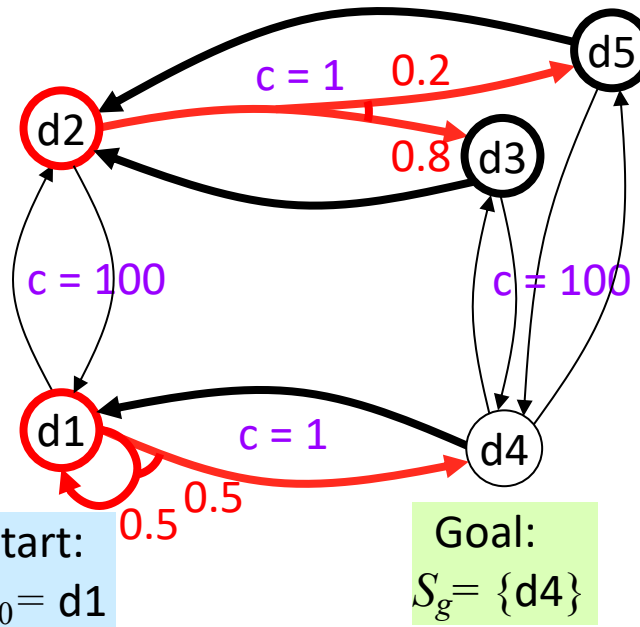
$$Q(d5, m54) = 100 + 0 = 100$$

$$V(d5) = 8; \pi(d5) = m52$$

$$r = \max(1^{7/8} - 1^{3/4}, 7 - 5, 8 - 6, 8 - 6) = 2$$

How long before $r \leq \eta$?

How long, if the “vertical” actions cost 10 instead of 100?



Discussion

- Policy iteration computes new π in each iteration; computes V^π from π
 - More work per iteration than value iteration
 - Needs to solve a set of simultaneous equations
 - Usually converges in a smaller number of iterations
- Value iteration
 - Computes new V in each iteration; chooses π based on V
 - New V is a revised set of heuristic estimates
 - Not V^π for π or any other policy
 - Less work per iteration: doesn't need to solve a set of equations
 - Usually takes more iterations to converge
- At each iteration, both algorithms need to examine the entire state space
 - Number of iterations polynomial in $|S|$, but $|S|$ may be quite large
- Next: use search techniques to avoid searching the entire space

Requires acyclic Σ

AO*

not in book

AO* (Σ, s_0, S_g, V_0)
 global $\pi \leftarrow \emptyset$; global $V(s_0) \leftarrow V_0(s_0)$
 global *Envelope* $\leftarrow \{s_0\}$ // generated states
 while $\text{leaves}(s_0, \pi) \setminus S_g \neq \emptyset$ do
 select $s \in \text{leaves}(s_0, \pi) \setminus S_g$
 for all $a \in \text{Applicable}(s)$
 for all $s' \in \gamma(s, a) \setminus \text{Envelope}$ do
 $V(s') \leftarrow V_0(s')$; add s' to *Envelope*

Bellman-Update(s)
 $v_{\text{old}} \leftarrow V(s)$
 for every $a \in \text{Applicable}(s)$ do
 $Q(s, a) \leftarrow \text{cost}(s, a) + \sum_{s' \in S} \text{Pr}(s'|s, a) V(s')$
 $V(s) \leftarrow \min_{a \in \text{Applicable}(s)} Q(s, a)$
 $\pi(s) \leftarrow \text{argmin}_{a \in \text{Applicable}(s)} Q(s, a)$
 return $|V(s) - v_{\text{old}}|$

AO-Update(s)

return π

no π -descendants in Z but s itself
 • ensures bottom-up updates

AO-Update(s)

$Z \leftarrow \{s\}$ // nodes that need updating

while $Z \neq \emptyset$ do

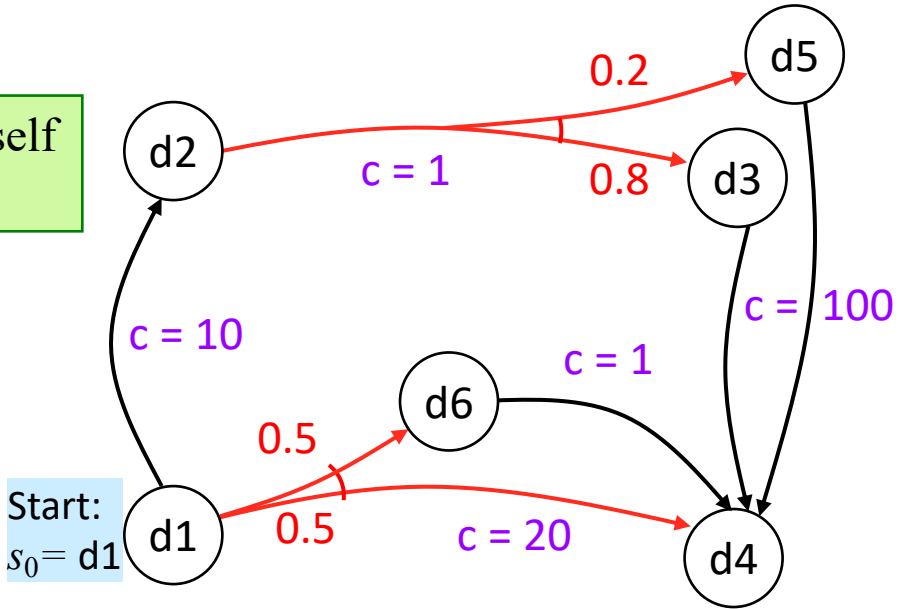
 select $s \in Z$ such that $\hat{\gamma}(s, \pi(s)) \cap Z = \{s\}$

 remove s from Z

 Bellman-Update(s)

$Z \leftarrow Z \cup \{s' \in \text{Envelope} \mid s \in \gamma(s', \pi)\}$

the states "just above" s

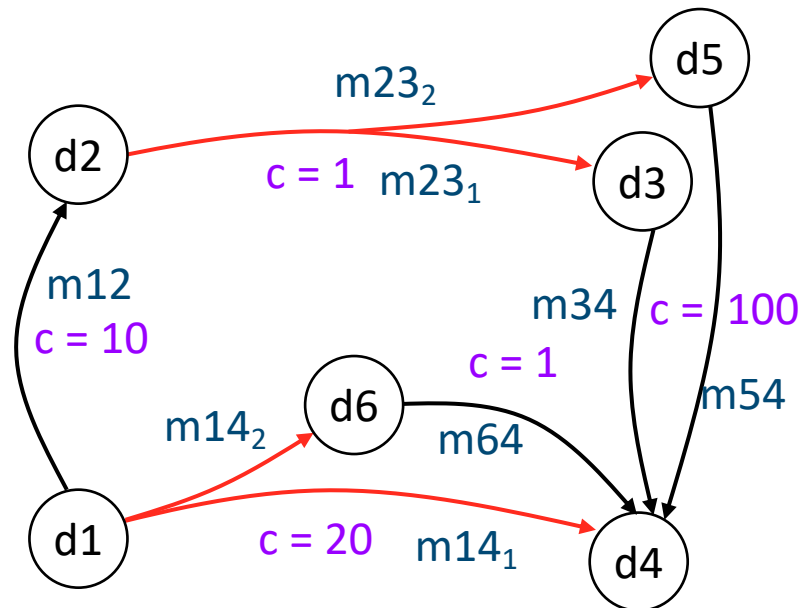
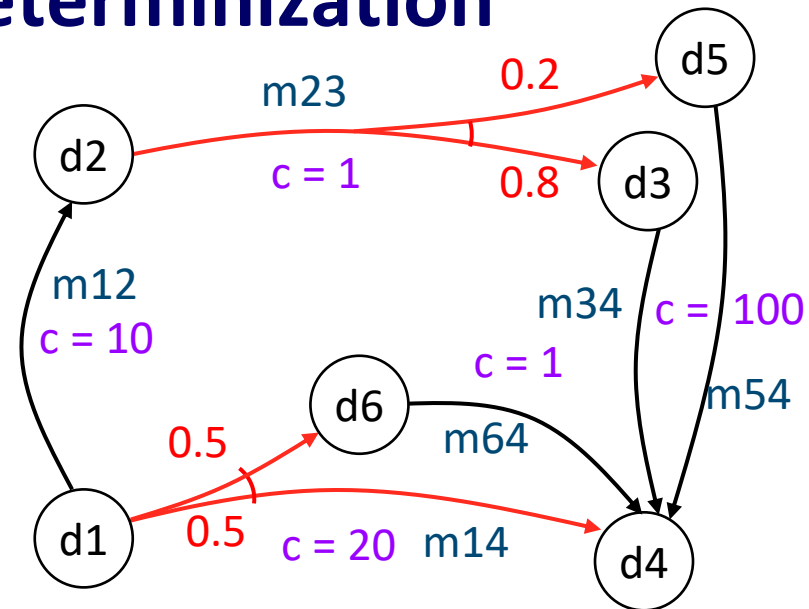


Example: $V_0(s) = 0$ for all s

Goal: $S_g = \{d4\}$

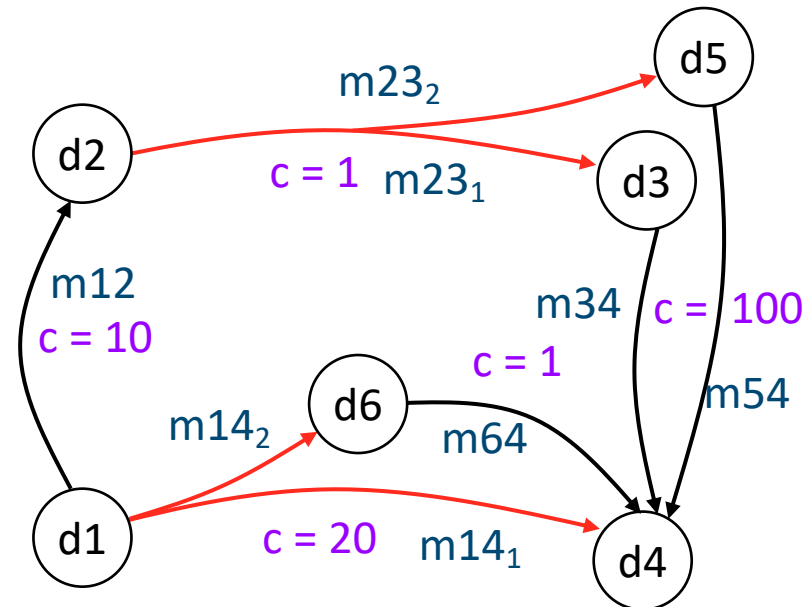
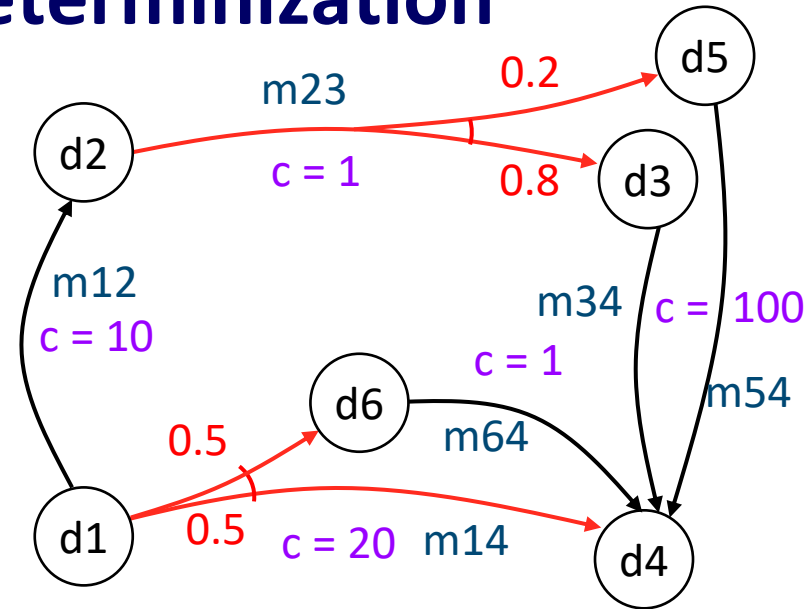
Heuristics through Determinization

- What to use for V_0 ?
 - One possibility: classical planner
 - Need to convert nondeterministic actions into something the classical planner can use
- *Determinize* the actions
 - Suppose $\gamma(s,a) = \{s_1, \dots, s_n\}$
 - $\text{Det}(s,a) = \{n \text{ actions } a_1, a_2, \dots, a_n\}$
 - $\gamma_d(s, a_i) = s_i$
 - $\text{cost}_d(s, a_i) = \text{cost}(s, a)$
- Classical domain $\Sigma_d = (S, A_d, \gamma_d, \text{cost}_d)$
 - S = same as in Σ
 - $A_d = \bigcup_{a \in A, s \in S} \text{Det}(s, a)$
 - γ_d and cost_d as above



Heuristics through Determinization

- Suppose we want $V_0(s)$
- Call classical planner on (Σ_d, s, S_g)
 - Get plan $p = \langle a_1, a_2, \dots, a_n \rangle$
 - Goes through states $\langle s, s_1, \dots, s_n \rangle$
 - $s_1 = \gamma(s, a_1), s_2 = \gamma(s_1, a_2), \dots$
 - Return $V_0(s) = \text{cost}(p) = \sum_i \text{cost}(a_i)$
- If the classical planner always returns optimal plans, then V_0 is admissible
- Outline of proof:
 - Let π be a safe solution in Σ
 - Every acyclic execution of π corresponds to a solution plan p' in Σ_d
 - Must have $\text{cost} \geq V_0(s)$
 - Otherwise the classical planner would have chosen p' instead of p



Σ may be either cyclic or acyclic

LAO*

LAO*(Σ, s_0, S_g, V_0)

not in book

global $\pi \leftarrow \emptyset$; global $V(s_0) \leftarrow V_0(s_0)$
 global $Envelope \leftarrow \{s_0\}$ // generated states

loop

if $leaves(s_0, \pi) \subseteq S_g$ then return π

select $s \in leaves(s_0, \pi) \setminus S_g$

for all $a \in Applicable(s)$

for all $s' \in \gamma(s, a) \setminus Envelope$ do

$V(s') \leftarrow V_0(s')$; add s' to $Envelope$

LAO-Update(s)

return π

All π -ancestors of s in $Envelope$

LAO-Update(s)

$Z \leftarrow \{s\} \cup \{s' \in Envelope \mid s \in \hat{\gamma}(s', \pi)\}$

loop

$r \leftarrow \max_{s \in Z} Bellman-Update(s)$

if $leaves(s_0, \pi)$ changed or $r \leq \eta$ then break

Asynchronous value iteration, restricted to Z

Bellman-Update(s)

$v_{old} \leftarrow V(s)$

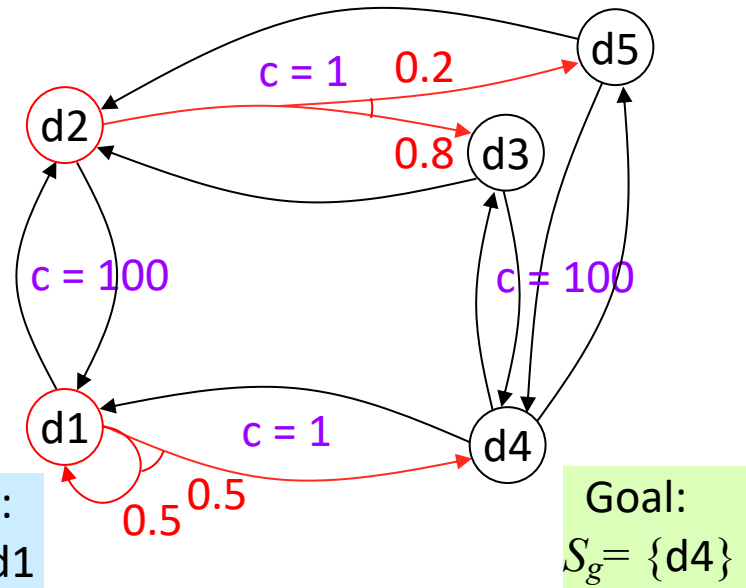
for every $a \in Applicable(s)$ do

$Q(s, a) \leftarrow cost(s, a) + \sum_{s' \in S} Pr(s'|s, a) V(s')$

$V(s) \leftarrow \min_{a \in Applicable(s)} Q(s, a)$

$\pi(s) \leftarrow \operatorname{argmin}_{a \in Applicable(s)} Q(s, a)$

return $|V(s) - v_{old}|$



Example: $V_0(s) = 0$ for all s

LAO* Example

1st iteration of main loop:

Expand d1: add d2 and d4 to *Envelope*

Call LAO-Update(d1)

π is empty, so $Z = \{d1\}$

Iteration 1:

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(0) + \frac{1}{2}(0)) = 1$$

$$V(d1) = 1; \pi(d1) = m14; r = V(d1) - 0 = 1$$

Iteration 2:

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1) + \frac{1}{2}(0)) = 1\frac{1}{2}$$

$$V(d1) = 1\frac{1}{2}; \pi(d1) = m14; r = 1\frac{1}{2} - 1 = \frac{1}{2}$$

Iteration 3:

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1\frac{1}{2}) + \frac{1}{2}(0)) = 1\frac{3}{4}$$

$$V(d1) = 1\frac{3}{4}; \pi(d1) = m14; r = 1\frac{3}{4} - 1\frac{1}{2} = \frac{1}{4}$$

$$\eta = 0.2$$

$$V_0(s) = 0$$

for all s

Iteration 4:

$$Q(d1, m12) = 100 + 0 = 100$$

$$Q(d1, m14) = 1 + (\frac{1}{2}(1\frac{3}{4}) + \frac{1}{2}(0)) = 1\frac{7}{8}$$

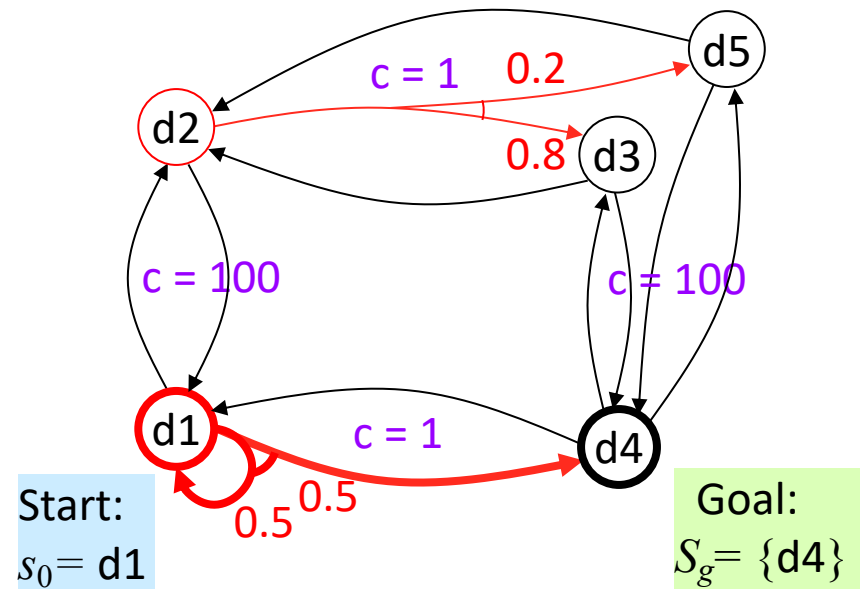
$$V(d1) = 1\frac{7}{8}; \pi(d1) = m14; r = \frac{1}{8} \leq \eta$$

LAO-Update returns

2nd iteration of main loop:

$$\text{leaves}(\pi) = \{d4\} \subseteq S_g$$

return π



Skipping Ahead

- Skipping ILAO*, HDP, LDFS_a, LRTDP, SLATE
 - I'll come back to these if there's time

Planning and Acting

Run-Lookahead(Σ, s_0, S_g)

$s \leftarrow s_0$

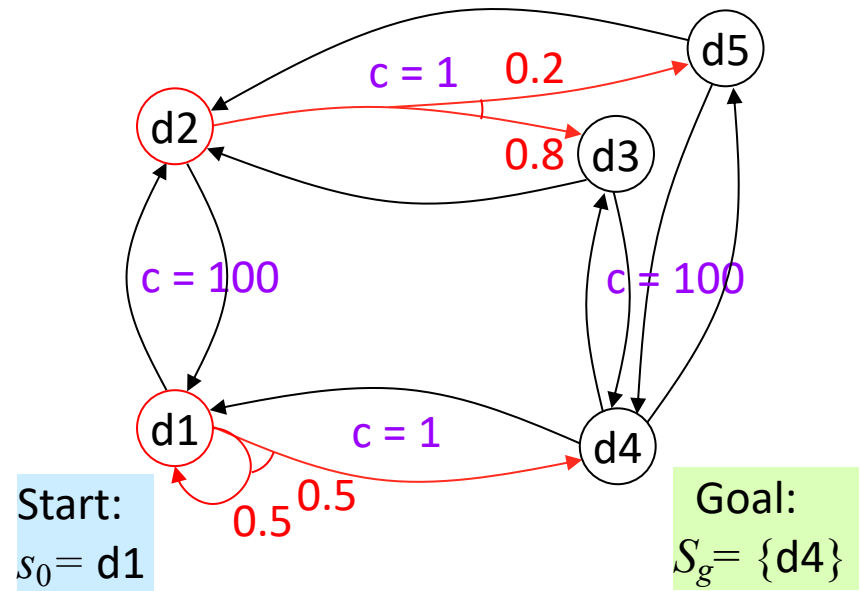
while $s \notin S_g$ and $\text{Applicable}(s) \neq \emptyset$ do

$a \leftarrow \text{Lookahead}(s, \theta)$

 perform action a

$s \leftarrow$ observe resulting state

- Same as in Chapter 2, except $s = \xi$
 - Could use $s \leftarrow$ abstraction of ξ as in Chapter 2
- Could also use Run-Lazy-Lookahead or Run-Concurrent-Lookahead



- What to use for Lookahead?
 - AO*, LAO*, ...
 - Modify to search part of the space
 - Classical planner running on determinized domain
 - Stochastic sampling algorithms

Planning and Acting

Run-Lookahead(Σ, s_0, S_g)

$s \leftarrow s_0$

while $s \notin S_g$ and $\text{Applicable}(s) \neq \emptyset$ do

$a \leftarrow \text{Lookahead}(s, \theta)$

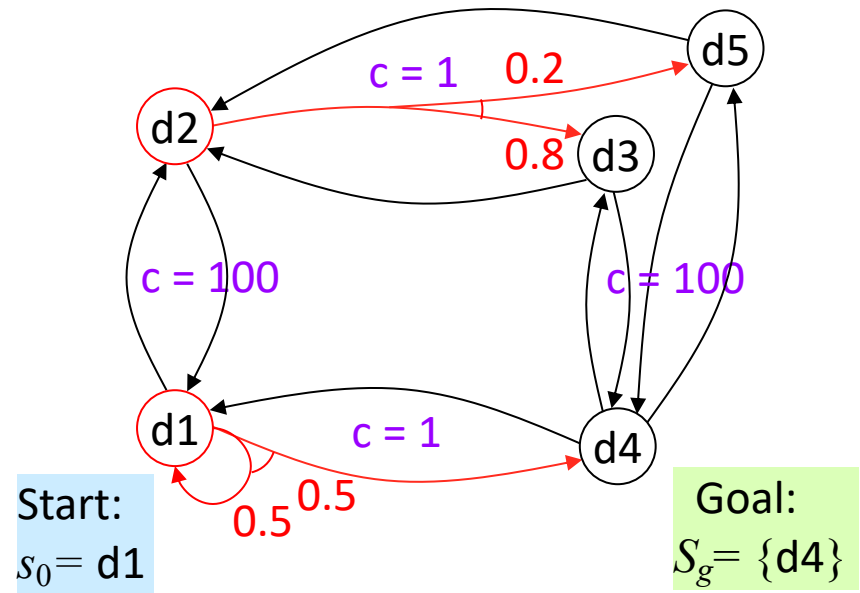
 perform action a

$s \leftarrow$ observe resulting state

- If Lookahead = classical planner on determinized domain

\Rightarrow FS-Replan (Chapter 5)

- Problem: Forward-search may choose a plan that depends on low-probability outcome
- RFF algorithm (see book) attempts to alleviate this



FS-Replan (Σ, s, S_g)

$\pi_d \leftarrow \emptyset$

while $s \notin S_g$ and $\text{Applicable}(s) \neq \emptyset$ do

 if π_d undefined for s then do

$\pi_d \leftarrow \text{Forward-search}(\Sigma_d, s, S_g)$

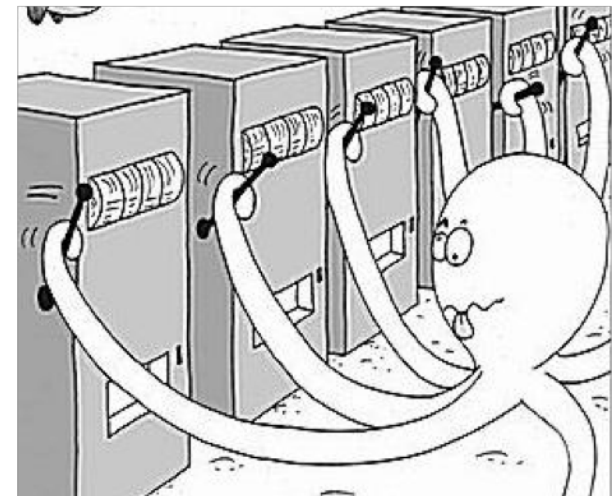
 if $\pi_d = \text{failure}$ then return failure

 perform action $\pi_d(s)$

$s \leftarrow$ observe resulting state

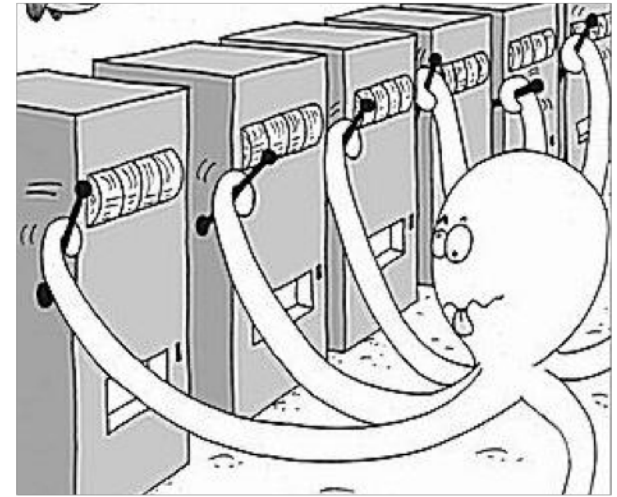
Multi-Arm Bandit Problem

- Statistical model of sequential experiments
 - Name comes from a traditional slot machine (one-armed bandit)
- Multiple actions a_1, a_2, \dots, a_n
 - Each a_i provides a reward from an unknown (but stationary) probability distribution p_i
 - Objective: maximize expected utility of a sequence of actions
- Exploitation vs exploration dilemma:
 - **Exploitation**: choose action that has given you high rewards in the past
 - **Exploration**: choose action that you don't know much about, in hopes that it might produce a higher reward



UCB (Upper Confidence Bound) Algorithm

- Assume all rewards are between 0 and 1
 - If they aren't, normalize them
- For each action a_i , let
 - r_i = average reward you've gotten from a_i
 - t_i = number of times you've tried a_i
 - $t = \sum_i t_i$



loop

if there are one or more actions that you haven't tried

then choose an untried action a_i at random

else choose an action a_i that has the highest value of $r_i + \sqrt{2(\ln t)/t_i}$

perform a_i

update r_i, t_i, t

UCT Algorithm

- Recursive UCB computation to compute $Q(s,a)$
- Anytime algorithm: call repeatedly until time runs out
 - Then choose action $\operatorname{argmin}_a Q(s,a)$

UCT(s, h)

if $s \in S_g$ then return 0

if $h = 0$ then return $V_0(s)$

if $s \notin \textit{Envelope}$ then do

 add s to *Envelope*

$n(s) \leftarrow 0$

 for all $a \in \textit{Applicable}(s)$ do

$Q(s,a) \leftarrow 0$; $n(s,a) \leftarrow 0$

$\textit{Untried} \leftarrow \{a \in \textit{Applicable}(s) \mid n(s,a) = 0\}$

 if $\textit{Untried} \neq \emptyset$ then $\tilde{a} \leftarrow \textit{Choose}(\textit{Untried})$

 else $\tilde{a} \leftarrow \operatorname{argmin}_{a \in \textit{Applicable}(s)} \{Q(s,a) - C \times [\log(n(s))/n(s,a)]^{\frac{1}{2}}\}$

$s' \leftarrow \textit{Sample}(\Sigma, s, \tilde{a})$

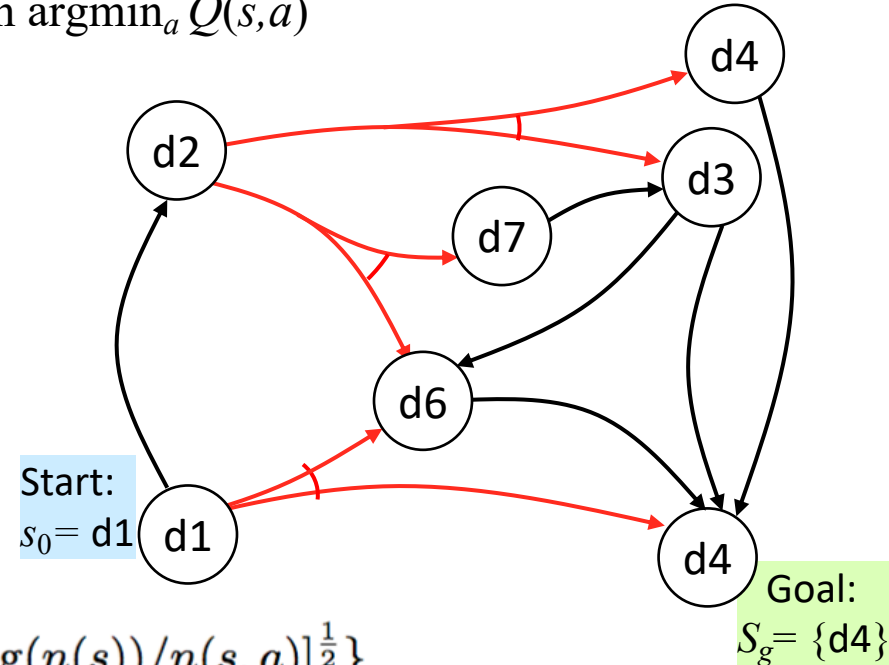
$\textit{cost-rollout} \leftarrow \textit{cost}(s, \tilde{a}) + \textit{UCT}(s', h - 1)$

$Q(s, \tilde{a}) \leftarrow [n(s, \tilde{a}) \times Q(s, \tilde{a}) + \textit{cost-rollout}] / (1 + n(s, \tilde{a}))$

$n(s) \leftarrow n(s) + 1$

$n(s, \tilde{a}) \leftarrow n(s, \tilde{a}) + 1$

 return $\textit{cost-rollout}$



UCT as an Acting Procedure

- Suppose you don't know the probabilities and costs
- Suppose you can restart your actor as many times as you want
- Can modify UCT to be an acting procedure

➤ Use it to explore the environment

UCT(s, h)

if $s \in S_g$ then return 0

if $h = 0$ then return $V_0(s)$

if $s \notin Envelope$ then do

add s to *Envelope*

$n(s) \leftarrow 0$

for all $a \in Applicable(s)$ do

$Q(s, a) \leftarrow 0; n(s, a) \leftarrow 0$

$Untried \leftarrow \{a \in Applicable(s) \mid n(s, a) = 0\}$

if $Untried \neq \emptyset$ then $\tilde{a} \leftarrow Choose(Untried)$

else $\tilde{a} \leftarrow \operatorname{argmin}_{a \in Applicable(s)} \{Q(s, a) - C \times [\log(n(s))/n(s, a)]^{\frac{1}{2}}\}$

$s' \leftarrow \mathbf{Sample}(\Sigma, s, \tilde{a})$

$cost\text{-rollout} \leftarrow cost(s, \tilde{a}) + UCT(s', h - 1)$

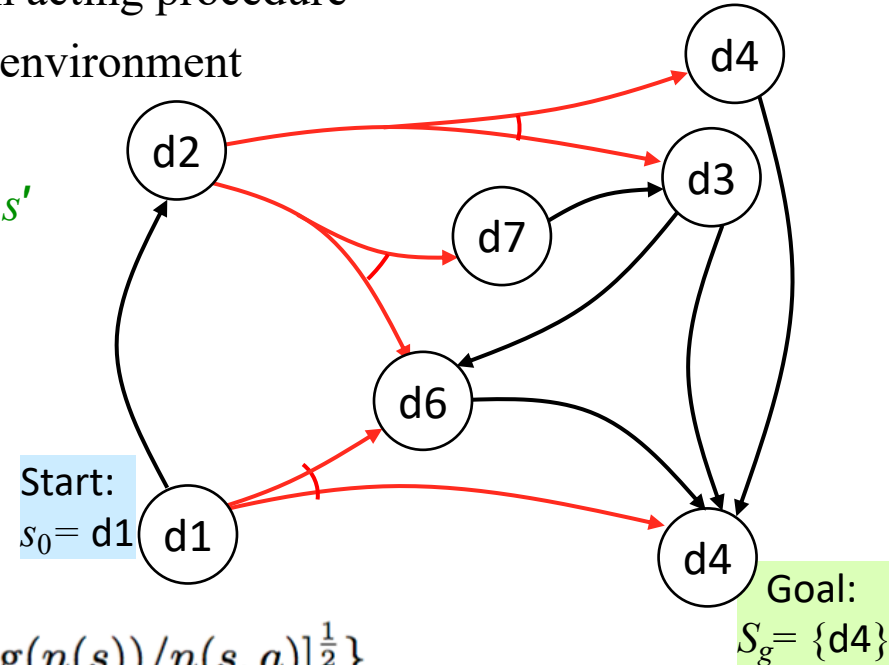
$Q(s, \tilde{a}) \leftarrow [n(s, \tilde{a}) \times Q(s, \tilde{a}) + cost\text{-rollout}] / (1 + n(s, \tilde{a}))$

$n(s) \leftarrow n(s) + 1$

$n(s, \tilde{a}) \leftarrow n(s, \tilde{a}) + 1$

return $cost\text{-rollout}$

perform a ; observe s'



UCT as a Learning Procedure

- Suppose you don't know the probabilities and costs
 - But you have an accurate simulator for the environment
- Run UCT multiple times in the simulated environment
 - Learn what actions work best

UCT(s, h)

if $s \in S_g$ then return 0

if $h = 0$ then return $V_0(s)$

if $s \notin Envelope$ then do

add s to *Envelope*

$n(s) \leftarrow 0$

for all $a \in Applicable(s)$ do

$Q(s, a) \leftarrow 0$; $n(s, a) \leftarrow 0$

$Untried \leftarrow \{a \in Applicable(s) \mid n(s, a) = 0\}$

if $Untried \neq \emptyset$ then $\tilde{a} \leftarrow Choose(Untried)$

else $\tilde{a} \leftarrow \operatorname{argmin}_{a \in Applicable(s)} \{Q(s, a) - C \times [\log(n(s))/n(s, a)]^{\frac{1}{2}}\}$

$s' \leftarrow \mathbf{Sample}(\Sigma, s, \tilde{a})$

$cost\text{-rollout} \leftarrow cost(s, \tilde{a}) + UCT(s', h - 1)$

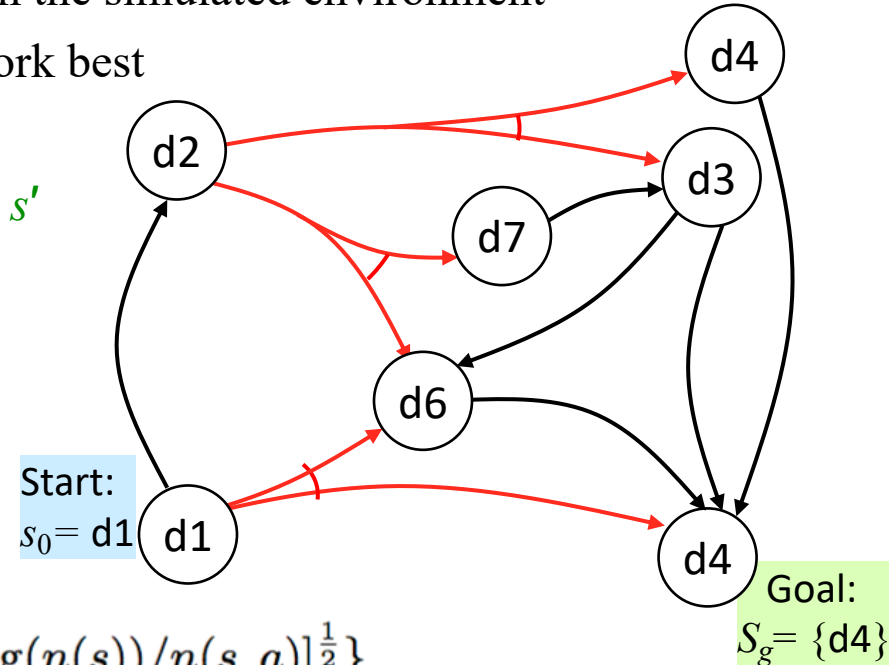
$Q(s, \tilde{a}) \leftarrow [n(s, \tilde{a}) \times Q(s, \tilde{a}) + cost\text{-rollout}] / (1 + n(s, \tilde{a}))$

$n(s) \leftarrow n(s) + 1$

$n(s, \tilde{a}) \leftarrow n(s, \tilde{a}) + 1$

return $cost\text{-rollout}$

simulate a ; observe s'

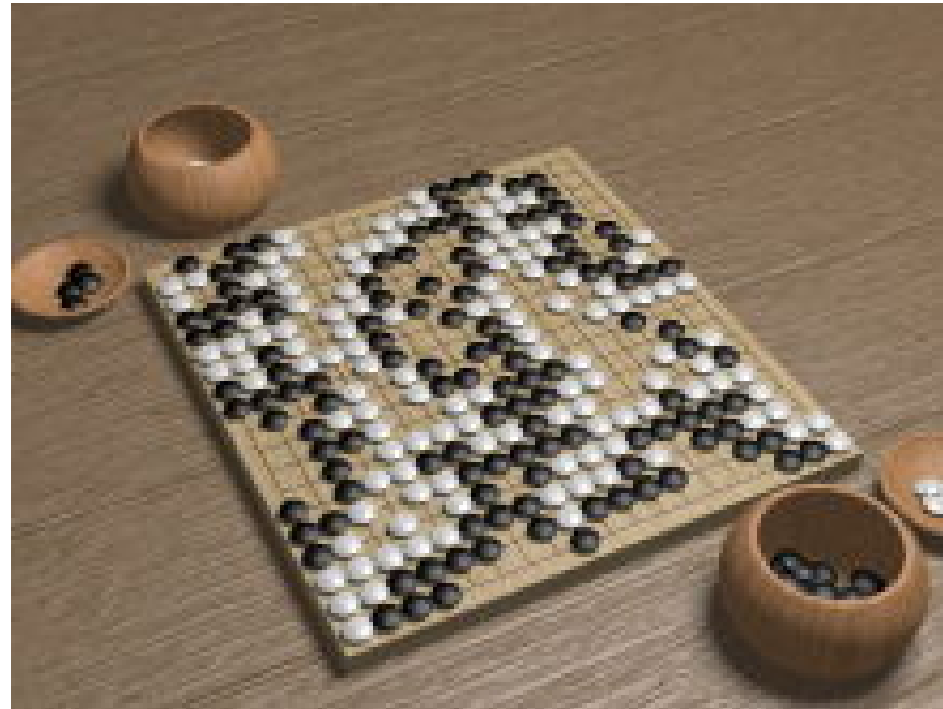


Start:
 $s_0 = d1$

Goal:
 $S_g = \{d4\}$

UCT in Two-Player Games

- Generate Monte Carlo rollouts using a modified version of UCT
- Main differences:
 - Instead of choosing actions that minimize accumulated cost, choose actions that maximize payoff at the end of the game
 - UCT for player 1 recursively calls UCT for player 2
 - Choose opponent's action
 - UCT for player 2 recursively calls UCT for player 1
- This produced the first computer programs to play go well
 - \approx 2008–2012
- Monte Carlo rollout techniques similar to UCT were used to train AlphaGo



Summary

- SSPs
- solutions, closed solutions, histories
- unsafe solutions, acyclic safe solutions, cyclic safe solutions
- expected cost, planning as optimization
- policy iteration
- value iteration (synchronous, asynchronous)
 - Bellman-update
- AO*, LAO*
- Planning and Acting
 - Run-Lookahead
 - RFF
- UCB, UCT